Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2016

JOÃO PEDRO GOMES RODRIGUES

MECANISMOS DE SEGURANÇA DE DADOS PARA PLATAFORMAS IOT

DATA SECURITY MECHANISMS FOR IOT PLATFORMS

Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2016

JOÃO PEDRO GOMES RODRIGUES

MECANISMOS DE SEGURANÇA DE DADOS PARA PLATAFORMAS IOT

DATA SECURITY MECHANISMS FOR IOT PLATFORMS

"*People who think they know everything are a great annoyance to those of us who do.*"

— Isaac Asimov

Universidade de Aveiro
2016

Departamento de Eletrónica,
Telecomunicações e Informática

JOÃO PEDRO GOMES RODRIGUES

MECANISMOS DE SEGURANÇA DE DADOS PARA PLATAFORMAS IOT

DATA SECURITY MECHANISMS FOR IOT PLATFORMS

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor João Paulo Silva Barraca, Professor auxiliar convidado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor André Ventura da Cruz Marnoto Zúquete, Professor auxiliar do Instituto de Engenharia Eletrónica e Informática de Aveiro da Universidade de Aveiro.

Dedico este trabalho à minha família pelo incansável apoio e motivação.

o júri / the jury

presidente / president

Prof. Doutor Rui Luís Andrade Aguiar
professor catedrático da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Miguel Filipe Leitão Pardal
professor auxiliar do Instituto Superior Técnico da Universidade de Lisboa

Prof. Doutor João Paulo Silva Barraca
professor auxiliar convidado da Universidade de Aveiro

agradecimentos / **acknowledgements**

Agradeço toda a ajuda e guia aos meus orientadores que tornaram possível a realização deste trabalho.

Palavras Chave arquitetura, segurança, privacidade, autorização, internet das coisas.

Resumo

As plataformas IoT (Internet das Coisas) existentes hoje em dia permitem que diversos dispositivos (“coisas”) com uma fraca capacidade de processamento, como sensores, estejam ligados à rede pública que é a Internet. São colhidos e partilhados dados do meio ambiente que nos rodeia permitindo-nos conhecer melhor o nosso mundo, agir de forma mais informada e garantir a funcionalidade de certos equipamentos. No entanto, atualmente, os dados são armazenados em claro nas plataformas IoT e podem ser acedidos na sua íntegra por terceiros causando diversos problemas de segurança como a falta de privacidade dos donos dos dados que não conseguem ter qualquer controlo sobre os seus dados produzidos. Como tal, é aqui apresentada uma modificação da arquitetura típica de uma plataforma IoT com o objetivo de adicionar a desejada segurança sobre os dados. Foram adicionadas entidades externas confiáveis de forma a que se conseguisse distribuir as chaves, os dados cifrados e a identificação do utilizador. Desta forma nenhuma entidade, por si só, terá o poder de pôr em causa a privacidade dos utilizadores. A entidade de processamento tem o papel de transformar dados criando anonimato. A entidade de certificação tem o papel de certificar código que irá efetuar essa transformação recorrendo a análise humana. A entidade de autorização tem o papel de permitir que o utilizador possa autorizar ou não todo este processo de acesso e obtenção dos seus dados por parte de terceiros.

Keywords architecture, security, privacy, authorization, internet of things.

Abstract

The IoT (Internet of Things) platforms existing today allow multiple devices ("things") with a low processing capacity, such as sensors, to be connected with the public network that is the Internet. Environmental data from our surrounding is collected and shared allowing us to better understand our world, act in a more informed manner, perform real-time monitoring of humans and ensure the functionality of certain equipment. However, currently, the data is stored in clear text in the IoT platforms and can be fully accessed by third parties causing many security problems as the lack of privacy of the owners of the data that cannot have any control over their production. As such, a modification of the typical IoT platforms architecture is presented here in order to add the desired security to the data. External trusted entities were added so that elements could be distributed such as the keys, the encrypted data and user identification. Thus, no entity alone has the power to undermine the privacy of users. The processing entity has the role of transforming data creating anonymity. The certification entity has the role of certifying the code that will make this transformation using human analysis. The authorization entity has the role of allowing the user to authorize or not all of this process of access and retrieval of their data by third parties.

Conteúdo

Lista de Figuras

Lista de Tabelas

Glossário

IoT	Internet of Things
SHA	Secure Hash Algorithm
AES	Advanced Encryption Standard
DES	Data Encryption Standard
MD5	Message-Digest algorithm 5
VPN	Virtual Private Network
TLS	Transport Layer Security
DNSSEC	Domain Name System Security Extensions
PIR	Private Information Retrieval
CLP	Controlador Lógico Programável
ZK	Zero-Knowledge
API	Application Programming Interface
RBAC	Role-Based Access Control
DTLS	Datagram TLS
DDoS	Distributed Denial of Service
RAM	Random Access Memory
PKI	Public Key Infrastructure
PaaS	Platform as a Service
IaaS	Infrastucture as a Service
TTP	Trusted Third Party
HMAC	Hash-based Message Authentication Code
MAC	Mandatory Access Control
DAC	Discretionary Access Control
ABE	Attribute-Based Encryption
CP-ABE	Ciphertext-Policy Attribute Based Encryption
KP-ABE	Key-Policy Attribute Based Encryption
M2M	Machine to Machine
DCapBAC	Distributed Capability-based Access Control
JSON	JavaScript Object Notation
CoAP	Constrained Application Protocol
IBE	Identity-Based Encryption
Idemix	Identity Mixer
PRE	Proxy Re-Encryption scheme
ETSI	European Telecommunications Standards Institute
DVD	Digital Versatile Disc
GPS	Global Positioning System
3GPP	3rd Generation Partnership Project
xDSL	Digital Subscriber Line
WLAN	Wireless Local Area Network
MSBF	M2M Service Bootstrap Function
MAS	M2M Authentication Server
SSL	Secure Sockets Layer
HTTP	HyperText Transfer Protocol
REST	REpresentational State Transfer
IV	Initialization Vector
CFB	Cipher FeedBack
SIM	Subscriber Identification Module
SCoT	Smart Cloud of Things
MQTT	Message Queue Telemetry Transport
UUID	Universally Unique IDentifier
SAML	Security Assertion Markup Language
OLED	Organic Light Emitting Diode

NAT	Network Address Translation
XML	eXtensible Markup Language
IdP	Identity Provider
PEM	Privacy Enhanced Mail
AES-NI	Advanced Encryption Standard New Instructions
KVM	Kernel-based Virtual Machine

capítulo 1

Introdução

Este capítulo irá introduzir o leitor ao tema tratado nesta dissertação.

O paradigma das comunicações Machine to Machine (M2M) é visto como um dos grandes impulsionadores da inovação durante os próximos anos. Graças a este paradigma é possível alcançar a visão da IoT, onde o nosso mundo real, os objetos com que interagimos e o ambiente que nos rodeia se tornam mais inteligentes. Sensores, ubiquamente disponíveis ao nosso redor permitem colher dados do nosso ambiente, das nossas interações e até mesmo de nós próprios. São exemplos os sistemas de monitorização de consumo energético, de localização remota, ou de monitorização intensiva de pacientes cardíacos. Tendo a informação presente em sistemas capazes de lidar com quantidades de informação massivas (*big data*), torna-se possível conhecer melhor o nosso mundo, ou agir de forma mais informada, melhorando o conforto ou mesmo os níveis de produtividade em diversas áreas.

Ao longo dos últimos anos tem sido desenvolvida, no Instituto de Telecomunicações, uma plataforma M2M multiprotocolar alinhada aos últimos desenvolvimentos e que tem sido utilizada por diversos projetos pilotos.

Um ponto pertinente deste tipo de sistemas é a segurança dos dados, quer na comunicação, quer no armazenamento e processamento. Em particular, como garantir que os dados viajam de forma confidencial, que é assegurada a integridade dos mesmos e que, em cada momento, é possível ter garantias relativas ao acesso à informação de uma entidade. Embora a lógica de serviço possa fornecer estas garantias, não existe proteção em caso de comprometimento dos sistemas por um atacante externo, ou simplesmente por um funcionário.

Atualmente as soluções para este tipo de problemas são maioritariamente resolvidas com contratos entre as diversas entidades, que estabelecem regras de utilização de dados, e cifras nas comunicações, não existindo soluções unificadas para os cenários de IoT. Esta dissertação tem como objetivo a criação de mecanismos de armazenamento seguro e confidencial de informação para cenários de IoT, onde o dono da informação tenha controlo sobre a mesma. Este sujeito é aquele cujas informações privadas são recolhidas e enviadas para a rede por dispositivos (sensores).

O utilizador, dono da informação recolhida e enviada para a rede, terá de ter controlo sobre a informação e autorizar expressamente que determinada entidade aceda aos seus dados produzidos. Estes dados não podem fluir na sua íntegra para as entidades que os requisitem. Desta forma, a privacidade dos utilizadores seria posta em causa, o que não seria desejável. Assim, apenas dispositivos ou componentes que estejam sobre o controlo do dono da informação, podem aceder ao conteúdo dos dados por ele produzidos, na sua forma integral.

Esta dissertação está dividida em vários capítulos. No Capítulo 2 o leitor será introduzido ao ambiente tratado nesta dissertação: a IoT. No Capítulo 3 está efetuada uma análise sobre o estado da arte atual, estudos, trabalhos e sistemas que foram efetuados nesta área ou áreas adjacentes à aplicação ou análise de problemas de segurança em ambientes de IoT. No Capítulo 4 é efetuada uma abordagem inicial sobre os problemas de segurança que existem neste tipo de plataformas assim como uma análise de outros requisitos que devem ser respeitados na elaboração de uma arquitetura de segurança para plataformas IoT. No Capítulo 5 é efetuada a descrição da solução proposta para introduzir os mecanismos de segurança de dados necessários para tornar o sistema IoT seguro. No Capítulo 6 é apresentada a validação da implementação da arquitetura que foi efetuada contendo um exemplo de aplicação simples, mostrando medidas de desempenho e sobrecarga no tamanho dos dados armazenados, uma análise sobre a usabilidade do sistema e a apresentação de uma tabela demonstrando que nenhuma entidade, por si só, consegue pôr em perigo a privacidade do utilizador. No último capítulo, resumem-se conclusões tiradas e perspetiva-se trabalho futuro.

CAPÍTULO 2

CONTEXTUALIZAÇÃO

Este capítulo irá contextualizar o leitor para que perceba o ambiente em que esta dissertação se insere. O mundo do IoT, alguns padrões e a descrição da plataforma IoT que foi utilizada para efetuar este trabalho são aqui explicados.

2.1 A IOT

Esta dissertação tem como principal objetivo resolver certos tipos de problemas de segurança que hoje em dia existem na IoT. Para tal é necessário saber o que é a IoT [1], [67].

Hoje em dia, com a evolução, somos cada vez mais dependentes da tecnologia. O ser humano já pouco faz sem diversos tipos de tecnologias, quer a nível de *hardware* quer a nível de *software*. O mundo e o contexto em que vivemos já não seria possível caso não existissem, por exemplo, sistemas computacionais. Por sistemas computacionais refiromo-nos a computadores de secretária, computadores portáteis, telemóveis, *tablets* e todos aqueles equipamentos que possuem dentro de si um processador e memória. Através deste tipo de componentes é possível efetuar diversas ações que produzem resultados finais úteis para o seu utilizador e que, caso não tivessem sido processados por uma máquina deste género, seriam muito mais difíceis e morosos de obter. Isto para não falar da exatidão e precisão com que os resultados são apresentados ao utilizador de um sistema computacional.

Estes dispositivos encontram-se, hoje em dia, ligados a uma rede mundial chamada Internet. Através da Internet é possível que um sistema computacional comunique e partilhe informação com outro sistema computacional, desde que este também esteja ligado a esta rede de redes. A Internet é, então, uma rede enorme que permite que vários dispositivos interajam sem que haja a necessidade de estarem perto uns dos outros,

fisicamente. No entanto, estes dispositivos estão normalmente limitados a sistemas computacionais complexos, como computadores e telemóveis, que possuam imensos recursos de processamento como um processador, memória de massa, memória volátil, interfaces de entrada e saída e outros componentes opcionais como leitores de Digital Versatile Disc (DVD) e processamento gráfico.

O conceito da IoT foi inventado com a intenção de alargar o número de tipos de dispositivos que se podem ligar a uma rede (gigante, como a Internet que conhecemos, ou não). Assim, para além dos equipamentos mais complexos, também equipamentos, objetos ou "coisas" se poderiam também ligar a uma rede para que pudessem trocar informação, comunicar e interagir. "Coisas" como uma lâmpada, uma televisão, um contador de eletricidade, uma porta (ou janela) motorizada, um medidor de tensão, um robô industrial, um drone ou até um carro seriam capazes de trocarem dados, não só entre si, mas também com todos os outros dispositivos ligados à Internet, caso a rede IoT subjacente o permita. O principal objetivo é, portanto, ligar o maior número de "coisas" possíveis e construir uma rede onde seja possível obter/partilhar informação gerada pelos diferentes objetos que existem (aqueles cuja informação pode ser útil).

Para que tal aconteça é necessário construir uma arquitetura IoT que o permita. Contudo, existem à partida diversas questões sobre esta ideia que tiveram de ser resolvidas. Questões como: Como ligar tudo o que existe no planeta? Que tipo de comunicação sem fios poderia ser implementada em dispositivos? Que modificações teriam de ser efetuadas à infraestrutura da Internet já existente para que passe a suportar biliões de novos dispositivos? Como fornecer energia a estes dispositivos? O que desenvolver para que a implementação da IoT seja economicamente viável? A maioria destas questões eram problemáticas no passado, uma vez que, atualmente, já estão parcialmente resolvidas.

2.2 CASOS DE USO DA IOT

A IoT pode ser utilizada em inúmeros casos para que se possa melhorar a qualidade de vida do dia-a-dia das pessoas no mundo em que vivemos [6].

No contexto das cidades inteligentes, a IoT pode ser utilizada, por exemplo, para monitorização de parques de estacionamento tornando-os inteligentes: através de sensores colocados nos diversos locais de estacionamento, é possível obter informação da ocupação desse lugar, comunicando toda esta informação a todos os utilizadores do parque; monitorização de vibrações e condições do material utilizados em edifícios, pontes e monumentos históricos: também através de sensores de vibração que comunicam esta informação às respetivas entidades competentes para que possam tomar precauções

de segurança e reparo; monitorização de ruído urbano como forma de alertar para condições de ruído anormais na zona urbana de uma cidade: utilizam-se sensores de ruído espalhados em zonas urbanas de cidades ou zonas industriais; deteção de *smartphones* que estejam a utilizar WiFi [14] ou Bluetooth; medidas dos níveis de campos eletromagnéticos construídos por diversos dispositivos ao redor de uma área e utilizar esses dados para monitorização, numa cidade; iluminação inteligente das ruas: colocação de sensores de luminosidade perto da iluminação pública de forma a evitar gastos de energia elétrica desnecessários.

No contexto ambiental, é possível empregar as tecnologias IoT para implementar sistemas de controlo de poluição do ar, deteção de incêndios florestais ou até mesmo deteção de tremores de terra. Também existem casos de aplicação relacionados com a água: monitorização de água potável, níveis de poluição no mar, fugas de água, monitorização e controlo da subida do nível do mar/rio, entre outros.

No contexto residencial, pode-se aplicar a IoT de forma a tornar mais prática e mais exata as medidas produzidas por contadores/medidores de água, gás e eletricidade numa casa. Estes medidores teriam de ser adaptados para que se pudessem integrar numa determinada plataforma IoT. Todos estes dados estariam disponíveis para os fornecedores facilitando a recolha de dados para efeitos de faturação, assim como para efeitos de deteção de avarias, deteção de consumo anormal ou até para estudos que determinadas empresas desejem fazer sobre o grau de consumo de determinado serviço no país ou numa região. Ainda no contexto residencial, existem aplicações na área da domótica. Ao fornecer a possibilidade dos diversos eletrodomésticos, dispositivos, portas e janelas motorizadas, televisores, luzes, etc. de estarem em contacto entre si e com, por exemplo, o dispositivo móvel ou o computador de um residente, é possível implementar sistemas de casas inteligentes onde cada componente pode ser controlado à distância (através do uso da IoT) e/ou programado para efetuarem certas ações num determinado momento: por exemplo, 10 minutos antes de um residente chegar a casa, o dispositivo móvel poderia comunicar com uma central de aquecimento (ou um simples ar condicionado) para que este se ligasse. Ou, por exemplo, programar o dispositivo móvel para que enviasse um comando a todas as persianas caseiras para fecharem assim que começasse a ficar de noite. Todos estes mecanismos aumentam a qualidade de vida das pessoas facilitando tarefas que seriam cansativas e repetitivas.

No contexto de logística, a IoT pode ser utilizada para que seja possível monitorizar a qualidade de um produto que esteja em processo de distribuição. Poderiam ser monitorizadas vibrações, temperatura, aberturas de contentor, etc. de forma a oferecer garantia de qualidade de entrega aos clientes. Para além disso, é possível monitorizar as localizações de diversas carrinhas de entrega: seria instalado um sensor em cada veículo de distribuição, permitindo a sua monitorização e controlo em tempo real a partir de

uma central, ou até permitir a comunicação entre os diversos veículos (por exemplo táxis).

No contexto industrial poderia-se monitorizar a qualidade do ar, a temperatura, presença de ozono e até efetuar diagnósticos sobre eventuais mal funcionamentos de máquinas industriais, através da colocação de sensores nas mesmas.

Existem muitos outros contextos onde a IoT pode ser aplicada, como por exemplo, na agro-pecuária: controlo de níveis de açúcar das uvas em vinhas de forma a potenciar a qualidade do vinho, controlar condições climáticas com o objetivo de maximizar a produção de frutas e vegetais, irrigação seletiva de zonas secas para que não se desperdice água desnecessariamente, controlo de meteorologia em diversos locais do planeta onde as respetivas informações são enviadas para centrais, controlo sobre as localizações de animais através da aplicação de *chips*/sensores agarrados aos animais para que estes possam ser facilmente localizados caso se percam ou fujam.

Finalmente, o contexto da saúde também é importante. A presença de dispositivos medidores dos sinais vitais de pessoas idosas, bebés ou pessoas diagnosticadas com doenças cardíacas/respiratórias (entre outros) pode ser vital para que estas sobrevivam. Estar em constante monitorização pelos médicos (que podem receber alarmes caso algum sinal vital não esteja nos níveis desejados) oferece segurança a nível emocional ao doente e também permite que seja rapidamente socorrido, no pior caso. Para além de doenças e monitorização de sinais vitais, dispositivos detetores de queda podem ser também instalados, por exemplo, em idosos ou pessoas com deficiência para rápida intervenção no caso de uma queda.

2.3 CAMADAS DA IOT

Qualquer arquitetura/plataforma IoT é, tipicamente, constituída por diversas camadas. No caso mais simples e mais abstrato, podemos de dizer que existem sempre 3 camadas principais. São elas: a camada dos produtores, criadores ou medidores de dados, a camada do armazenamento ou gestão dos dados e a camada dos consumidores dos dados. A camada dos produtores trata de produzir e agregar os dados do contexto em que se encontra e enviá-los para armazenamento. Os produtores são geralmente sensores, no entanto, pode ser qualquer outro dispositivo que origine dados para que estes possam ser armazenados, mais tarde. A camada de armazenamento é a plataforma IoT em si. É a plataforma que recebe os dados e os trata de uma determinada forma. Este tratamento inclui o seu armazenamento persistente, localmente ou remotamente. Os consumidores são normalmente aplicações ou serviços que usam a plataforma IoT para aceder aos dados. Os consumidores, sempre que necessitam de obter dados, procedem à

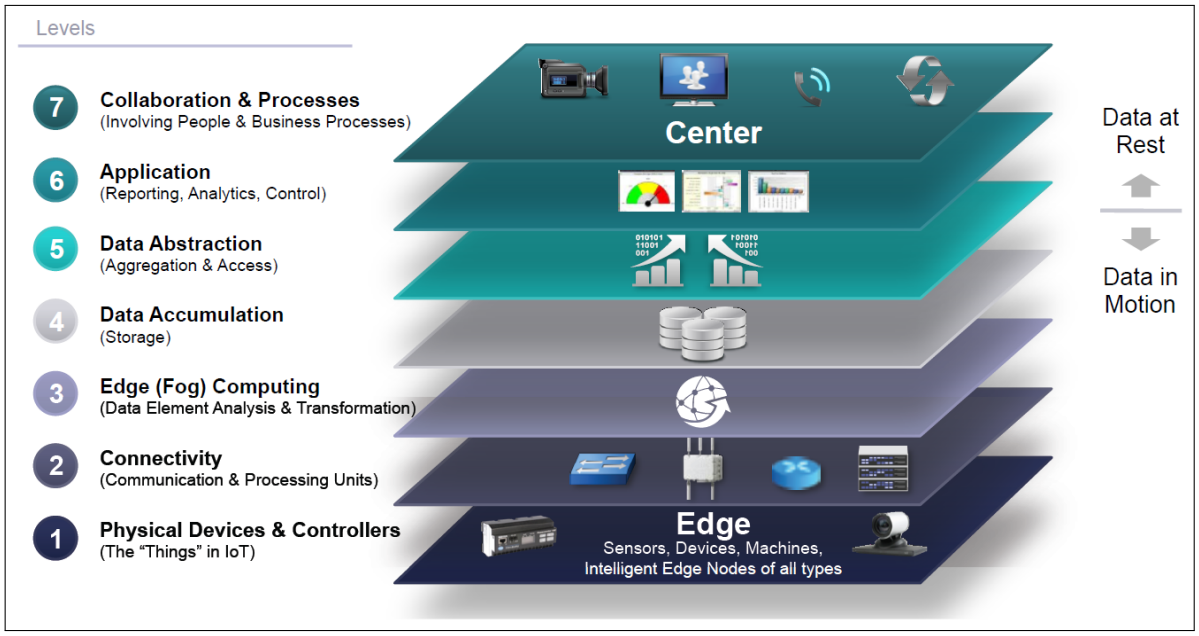

Figura 2.1: Modelo de referência para uma arquitetura IoT composto por camadas, retirado de [1]

construção de um pedido que enviam para a plataforma. A plataforma, caso aceite esse pedido, trata de obter os dados das bases de dados (ou outra forma de armazenamento) e reencaminhar para as aplicações/serviços requerentes. O fluxo dos dados é então, originado no produtor, enviado para a infraestrutura e consumido no nível mais alto.

Aumentando agora o nível de detalhe na explicação, na verdade uma plataforma IoT pode, e deve, ser constituída por muitas outras camadas. Isto porque num sistema IoT existem muitos tipos de dispositivos. É necessário que a plataforma esteja preparada para a heterogeneidade dos dispositivos, assim como para os diferentes meios de comunicação com o exterior que suportam. Para além disso, os dados produzidos pelos diferentes dispositivos não são todos do mesmos tipo, pelo que necessitam de ser processados de diferentes formas e não igualmente pela plataforma, pois a aplicação de uma plataforma IoT pode ser extremamente variada, como foi referido na Secção 2.2. Adicionalmente, os dados podem, opcionalmente, ser armazenados em diferentes localizações (diferentes bases de dados, locais geográficos). Tal acontece, normalmente, devido a várias razões: (1) demasiados dados para estarem num só lugar; (2) vantagens económicas; (3) questões legais: cada país tem a sua própria lei que define certas regras para o armazenamento de informação. Por exemplo, é ilegal, em certos países, que determinados tipos de agregados de informação sensível sejam armazenados num serviço público sem a aplicação correta de mecanismos de segurança [8], [13]. Finalmente, as diferentes aplicações/serviços (consumidores) que acedem aos dados são heterogéneas, podendo estas funcionar de forma diferente do normal e processar os dados de forma diferente. Uma vez que

dispositivos de tipos diferentes podem originar tipos de dados diferentes, comunicando-os de forma diferente, necessitando de políticas de armazenamento diferente e sendo acedidos e processados de forma diferente, surgiu a necessidade de criar um modelo de referência, que deve ser a base de qualquer infraestrutura IoT, baseado em diferentes camadas de modo a potenciar o suporte a toda a heterogeneidade, apoiar a escalabilidade e manter a simplicidade. Na Figura 2.1, está representado esse modelo. Como foi mencionado anteriormente, o fluxo de dados efetua-se normalmente da camada mais baixa (camada 1 - produtores) para a camada mais alta (camada 7 - consumidores), contudo, na realidade, a bidirecionalidade deve existir para que seja possível implementar mecanismos de controlo, gestão de dispositivos e manutenção de toda a infraestrutura.

A primeira camada é composta por controladores de dispositivos, que podem controlar outros e agregar dados, e pelos dispositivos em si. Estes dispositivos são os produtores de dados. Podem ser sensores, máquinas, nós inteligentes ou qualquer outro dispositivo que produza informação para que esta seja armazenada e comunicada, eventualmente, às aplicações (localizadas na camada superior). São as chamadas "*things*" ("coisas") e o seu *hardware*, funcionamento, tamanho, origem e localização pode variar a um nível extremo. Um carro pode ser uma "coisa" produtora de informação como, por exemplo, imagens ao seu redor. Um sensor Global Positioning System (GPS) pode também ser uma "coisa" que produz informação de localização. Os componentes nesta camada devem ser capazes de gerar dados, serem controlados através da rede, efetuar conversões analógico-digital, caso tal seja necessário. O modelo de referência menciona, ainda, a necessidade de segurança que tem que existir a este nível: os dispositivos têm de ser seguros: resistentes a modificações e a invasões físicas/virtuais de forma a que seja possível produzir e enviar dados fiáveis e sem nenhum tipo de alterações maliciosas. Este tipo de métodos é importante de forma a que o conteúdo gerado pelo dispositivo esteja em segurança no próprio dispositivo, evitando certos ataques que afetem a privacidade ou a integridade e correção da informação colhida.

A segunda camada é composta por unidades de processamento e comunicação. Nesta camada estão componentes que efetuam a ligação entre os componentes da camada inferior (os dispositivos produtores) e a rede e entre a rede e a camada superior (camada 3) na plataforma. Toda a parte de comunicação e passagem de dados através da rede (pública ou não, já existente ou não) é efetuada nesta camada. Nesta camada incluem-se *gateways*, *routers*, *switches* entre outros equipamentos de rede e comunicação. Os componentes nesta camada devem assegurar a comunicação fiável dos dados, o suporte de vários protocolos assim como suporte à tradução ente eles (de forma a combater a possível heterogeneidade), *switching*, *routing* e segurança na transmissão dos dados a nível da rede (segurança a nível de *hardware* e a nível dos protocolos na passagem de informação).

A terceira camada é composta por componentes de pré-processamento, análise e transformação dos dados que chegam da rede à infraestrutura IoT. Este pré-processamento tem de ser efetuado aos dados devido à variedade de tipos de dados que pode chegar até esta camada. Estes elementos tratam de homogeneizar e traduzir os dados para que o seu formato seja percebível pela camada acima desta podendo, eventualmente, serem acrescentados meta-dados e muitas outras informações de forma a que possam estar preparados para que possam ser mais tarde armazenados e recuperados através de pesquisas. Nesta camada os dados são analisados e pré-processados um a um não existindo qualquer conceito de estado ou de sessão por parte dos componentes aqui presentes. O principal pré-processamento de dados efetuado nesta camada inclui: filtragem de dados, a sua limpeza, a agregação de várias dados do mesmo tipo, inspeção do conteúdo dos pacotes identificando anomalias ou o tipo de dados que foi transmitido pelos dispositivos produtores, aplicação de limiares (*thresholds*) e geração de eventos para outras camadas como forma de aviso ou apenas de informação sobre a chegada de um determinado formato de dados. Processos de cifra e gestão de segurança podem ser aplicados nesta camada, antes dos dados serem encaminhados para camadas superiores.

A quarta camada é composta por componentes armazenadores de dados que já foram produzidos pelos produtores presentes na camada 1, enviados pela rede cujas operações são garantidas pela camada 2, pré-processadas pelos componentes de análise e transformação de mensagens/protocolos feitos na camada 3. Chegando à quarta camada, os dados passam de estar em movimento para estarem parados. Esta camada de acumulação de dados acumula os dados e depois armazena-os. Este tipo de armazenamento pode ser feito de várias formas (armazenamento em disco, em memória volátil (Random Access Memory (RAM)), etc.), o tipo de armazenamento pode ser variado (bases de dados, *cloud*, sistema de ficheiros, etc.) e os dados podem necessitar de serem agregados de várias fontes e combinados/reprocessados de alguma forma para eficiente armazenamento. Todas estas operações são feitas nesta camada juntamente com o armazenamento em si mesmo. Aqui, os dados deixam de estar em constante movimento, sujeitos a transformações, e passam a estar parados (estáticos) para futuro acesso. Nesta camada podem, então, ser executadas diferentes ações: conversão de dados em pacotes para dados que possam ser armazenados em bases de dados relacionais, efetuar a transição da computação baseada por eventos (efetuada desde a primeira camada até esta) para computação baseada em *queries* (efetuada desde esta camada até à última camada), conversão de dados em movimento para dados estáticos, dinamicamente, reduzir a quantidade de dados através de filtros adicionais, efetuar armazenamento seletivo, agregação e acumulação de dados provenientes da camada 3 (pré-processamento).

Uma vez que os dados estão, agora, em modo de repouso e armazenados, é necessário possibilitar às camadas superiores (aquelas que irão consumir os dados aqui armazenados)

interfaces de acesso aos mesmos. A quinta camada tem como objetivo agregar os dados armazenados para que estes possam ser expostos através de uma interface para os consumidores. Uma vez que os dados podem estar armazenados em inúmeros lugares do planeta, repartidos em pedaços por diferentes bases de dados ou numa única base dados (heterogeneidade no armazenamento), é necessário que esta camada providencie mecanismos de abstração sobre o acesso aos dados. Esta abstração deve suportar a reconciliação de múltiplos dados de diferentes origens oferecendo simplicidade no acesso às aplicações, assegurar consistência semântica entre as diferentes origens, assegurar que os dados estão completos e com boa qualidade para que estes sejam transmitidos para as camadas de aplicação localizadas em camadas superiores, filtrar, selecionar, projetar, formatar, esquematizar e criar vistas sobre os dados para mostrar os dados aos consumidores da forma que eles estão à espera de os receber, proteger os dados com controlo de acesso (autorização, autenticação) e providenciar às aplicações um acesso rápido aos dados efetuando a indexação da informação. Todos estes mecanismos simplificam o modo como as aplicações devem ser desenvolvidas para acederem aos dados criando uma abstração sobre o acesso que resolve o problema da heterogeneidade que existe ao nível aplicacional e de serviços consumidores em camadas superiores.

A sexta camada é composta por aplicações ou serviços consumidoras que têm de aceder aos dados através da camada de abstração abaixo. As aplicações efetuam *queries* e obtém respostas com a informação que pretendem para que esta possa ser representada. Estas aplicações podem ser de monitorização: por exemplo, monitorizar a temperatura presente dentro de um *datacenter* obtendo informação reportada por sensores de temperatura existentes no edifício/infraestrutura; podem ser aplicações de controlo: através do clique na interface gráfica da mesma ou através de eventos programados, uma aplicação de controlo (móvel ou não) pode enviar comandos para dispositivos inteligentes como luzes ou portas e janelas motorizadas; aplicações de caráter empresarial: informações úteis ou críticas obtidas através de outros dispositivos como sensores, medidores, etc. que sejam vitais para que sejam tomadas decisões a nível empresarial ou financeiro. Estas aplicações são, tipicamente, utilizadas por seres humanos. É aqui que os consumidores visualizam os dados que foram recolhidos pelos sensores, comunicados através da rede, transformados de forma a tornarem-se armazenáveis e armazenados.

Os utilizadores das aplicações, ao receberem os dados corretos através das diferentes interfaces, geralmente, gráficas das diferentes aplicações comunicam agora e criam a última (sétima) camada da IoT denominada camada de colaboração e processos de negócio. Depois de analisar a informação recolhida, são tomadas decisões e são criadas ações que envolvem pessoas e processos. Estas ações não são criadas por apenas uma pessoa mas sim por várias. Assim, é necessário existir colaboração e comunicação entre

as várias pessoas de forma a que possam produzir trabalho de melhor qualidade.

2.4 ARQUITETURA ETSI PADRÃO PARA COMUNICAÇÕES M2M

Para que seja possível construir uma arquitetura IoT, é necessária a presença das várias camadas mencionadas e descritas na secção anterior. A maioria dos componentes presentes nestas camadas são diferentes sistemas computacionais. Estas máquinas necessitam de comunicar entre si. A forma como elas comunicam, os protocolos que usam, as estruturas de dados que cada componente deve ter presentes e as interfaces que deve mostrar para o exterior foi padronizada pelo European Telecommunications Standards Institute (ETSI), um instituto que define padrões europeus [2].

O ETSI apresenta, então, uma arquitetura pensada e detalhada que procura ser flexível, escalável e funcional nas comunicações entre máquinas (M2M). O ETSI desenhou-a para que possa fazer uso do protocolo IP (Internet Protocol).

2.4.1 ARQUITETURA

Na Figura 2.2 está representada a arquitetura padrão proposta. Esta arquitetura padrão e o modo como as várias entidades comunicam pode estar incluída e implementada numa plataforma IoT, devido à existência de comunicações entre máquinas. O esquema define dois domínios: o domínio dos dispositivos produtores de dados e *gateways* e o domínio da rede. Na Figura 2.1, onde estão representadas as diferentes camadas de uma infraestrutura IoT, o domínio dos dispositivos e dos *gateways* pode estar incluído na camada 1 e em parte da camada 2. A camada 2 também contempla a comunicação entre o dispositivo e a rede (podendo ser feita a inserção do *gateway* aqui), assim como a comunicação desde a rede até à plataforma IoT (que já não é do domínio do dispositivo/*gateway*, mas sim do domínio de rede). O domínio de rede pode estar incluído no que resta da camada 2 prolongando-se até à última (camada 7).

O domínio do dispositivo e do *gateway* é composto por vários elementos descritos de seguida.

O M2M Device (dispositivo M2M) é um dispositivo que pode, em alguns casos, executar aplicações M2M. Estes dispositivos, tipicamente, produzem os dados com o intuito de serem futuramente recebidos pelos consumidores. Estes dispositivos necessitam de se ligar à rede, podendo fazê-lo de duas formas: ligar-se diretamente à rede de acesso executando procedimentos de registo, autenticação, autorização, entre outros; ou

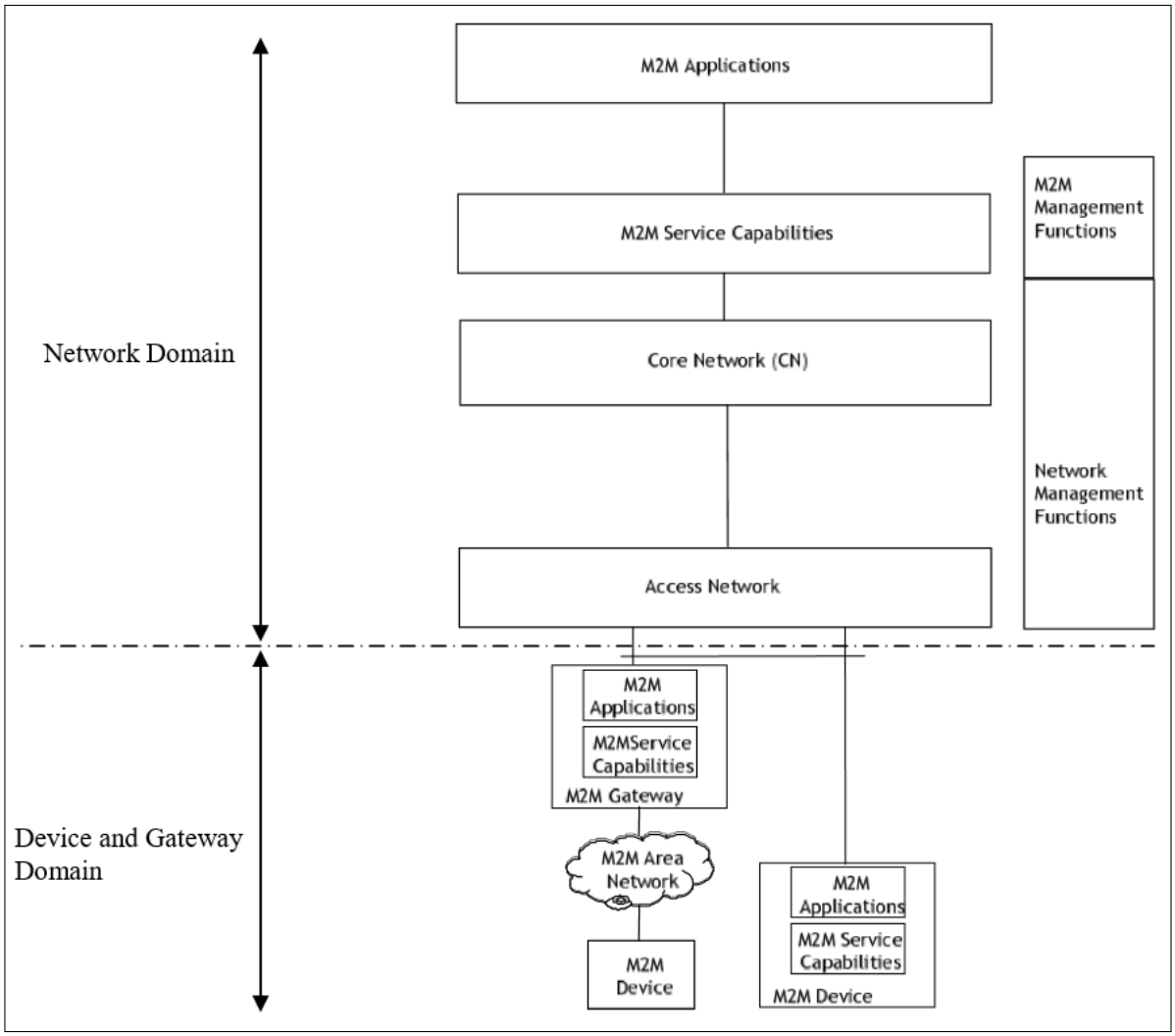

Figura 2.2: Arquitetura ETSI padrão para comunicações M2M, retirado de [2]

ligar-se utilizando um *gateway* como um *proxy*. Para que o dispositivo se possa ligar à rede tem de, primeiramente, ligar-se a um *gateway* através de uma rede de área M2M (M2M Area Network). O *gateway* atua como um *proxy* e encaminha as mensagens provenientes dos dispositivos a ele ligados para a rede de acesso presente no domínio de rede da arquitetura M2M. No caso da existência de um *gateway*, é este que trata de operações de registo, autenticação e autorização devido à comum falta de capacidade de processamento e armazenamento dos dispositivos que a ele se ligam (podendo ser em número variado). Normalmente, os dispositivos que necessitam de se ligar a um *gateway* apenas têm a função principal de medir e enviar dados, não sendo capazes de executar quaisquer aplicações M2M nem de suportar capacidades de serviço M2M inerentes à comunicação com o resto dos componentes da arquitetura M2M.

A M2M Area Network (rede de área M2M), é uma rede usada por um dispositivo para que se consiga ligar a um *gateway*. Zigbee [42] e Bluetooth [29] são exemplos de

tecnologias de comunicação em rede que podem ser utilizadas neste contexto.

Por fim, o *gateway* M2M, é um componente que possui todas as funcionalidades que um dispositivo M2M normal teria que suportar (execução de aplicações e suporte a capacidades de serviço M2M) caso o dispositivo M2M não necessitasse de estar ligado a um *gateway*, uma vez que teria de possuir capacidades de processamento e armazenamento acima do normal para efetuar todas as operações necessárias. Para além disso, o *gateway* também serve como agregador dos dados de vários dispositivos. Assim, um *gateway* pode fazer de *proxy* para muitos dispositivos de fraca capacidade de computação e comunicação. Isto é útil no caso do uso de vários sensores. Uma vez que estes apenas medem e efetuam processamento muito básico de dados (através de um micro-controlador), todos poderiam estar ligados a um *gateway* que encaminha os dados e fala pelos diversos dispositivos, ao invés de desperdiçar recursos na elaboração de dispositivos mais complexos para que, individualmente, pudessem interagir com a restante rede M2M.

O domínio da rede é composto por vários elementos descritos de seguida.

A Access Network (rede de acesso) é a rede utilizada pelos *gateways* M2M ou pelos dispositivos M2M para comunicar com a rede núcleo M2M. Exemplos de redes que podem ser aqui utilizadas são: Digital Subscriber Line (xDSL), satélite, Wireless Local Area Network (WLAN), entre outros. A rede de acesso poderia pertencer à camada 2 do modelo de referência para uma arquitetura IoT, presente na Figura 2.2.

A Core Network (rede nuclear), localizada logo acima da rede de acesso, é um local que, segundo o padrão, tem de fornecer, no mínimo, conectividade IP. No entanto pode fornecer muitos outros meios de ligação entre os diversos componentes, funções de controlo sobre toda a rede e sobre serviços, interligação com outras redes para que possa existir comunicação com o exterior à rede atual, caso haja necessidade para tal, e *roaming* a todos os dispositivos da rede.

O componente M2M Service Capabilities (capacidades de serviço M2M) localizado entre a rede nuclear e as aplicações M2M fornece determinadas funções M2M que devem ser suportadas pelos diversos dispositivos/*gateways* M2M, assim como pela própria rede. Estas funções M2M, expostas através de um conjunto de interfaces, são comuns e são partilhadas pelos diferentes pontos da arquitetura que necessitem de capacidades de serviço M2M, ou seja, que necessitem de executar operações/procedimentos relacionados com a comunicação M2M. Através das interfaces expostas pelos componentes com este tipo de capacidades, é possível simplificar e otimizar o desenvolvimento e a instalação das diferentes aplicações. Isto porque, os métodos incluídos nas capacidades de serviço, que já incluem toda a lógica de comunicação entre máquinas (ou rede, caso em que usa as funcionalidades da rede nuclear) necessária para que a interação seja bem sucedida e aceite pelo resto do sistema, já se encontram implementados e prontos a serem utilizados

através de uma Application Programming Interface (API). Quer a rede nuclear quer as capacidades de serviço M2M possuem determinadas funções que podem ser incluídas nas camadas 3, 4 e 5 do modelo de referência para uma arquitetura IoT, ilustrado pela Figura 2.2.

As M2M Applications (aplicações M2M) são aplicações ou serviços consumidoras de dados que utilizam as capacidades de serviço M2M como forma de comunicação com a rede nuclear do sistema. Esta comunicação existente entre as aplicações e a rede existe para que seja possível, entre outras coisas, obter dados que foram medidos pelos dispositivos produtores. As aplicações M2M podem-se incluir na camada 6 do modelo presente na Figura 2.2.

Para além destes componentes, ainda são necessários componentes de gestão que possuam algumas funções. As Network Management Functions (funções de gestão de rede) devem estar disponíveis numa arquitetura de comunicação M2M. O objetivo é efetuar toda a gestão da rede de acesso e nuclear incluindo supervisão, gestão de falhas, provisionamento, entre muitas outras operações de gestão de redes. As M2M Management Functions (funções de gestão de M2M) são todas aquelas funções que não estão envolvidas no fluxo de dados normal, mas são inevitáveis para gerir todos os componentes que estejam relacionados e necessitem de utilizar comunicações M2M. Toda a gestão de dispositivos e *gateways* é executada com recurso a funções deste tipo. Para além de toda a gestão dos componentes M2M, existe uma função denominada de M2M Service Bootstrap Function (MSBF). Esta função permite efetuar o provisionamento permanente de credenciais de segurança nos diversos componentes M2M. Estas credenciais são guardadas num local seguro denominado de M2M Authentication Server (MAS). As credenciais permanentes relacionadas com os dispositivos ou *gateways* M2M são guardadas num domínio seguro do próprio dispositivo. Todas estas credenciais são usadas para efeitos de autorização a acesso a serviços, sistemas de faturação e mútua autenticação e comunicações seguras entre os componentes de capacidades de serviço dos dispositivos/*gateway* M2M e os componentes de capacidades de serviço presentes na rede.

2.4.2 FLUXO DE EVENTOS

Na Figura 2.3 está representado, a alto nível, o fluxo de eventos relacionados com esta arquitetura M2M. Não será detalhado o fluxo, apenas irão ser mencionadas e descritas, a alto nível, as operações principais que são executadas.

As operações de provisionamento de credenciais de segurança é o primeiro passo a ser efetuado para que seja possível para um dispositivo/*gateway* M2M entrar e registar-se

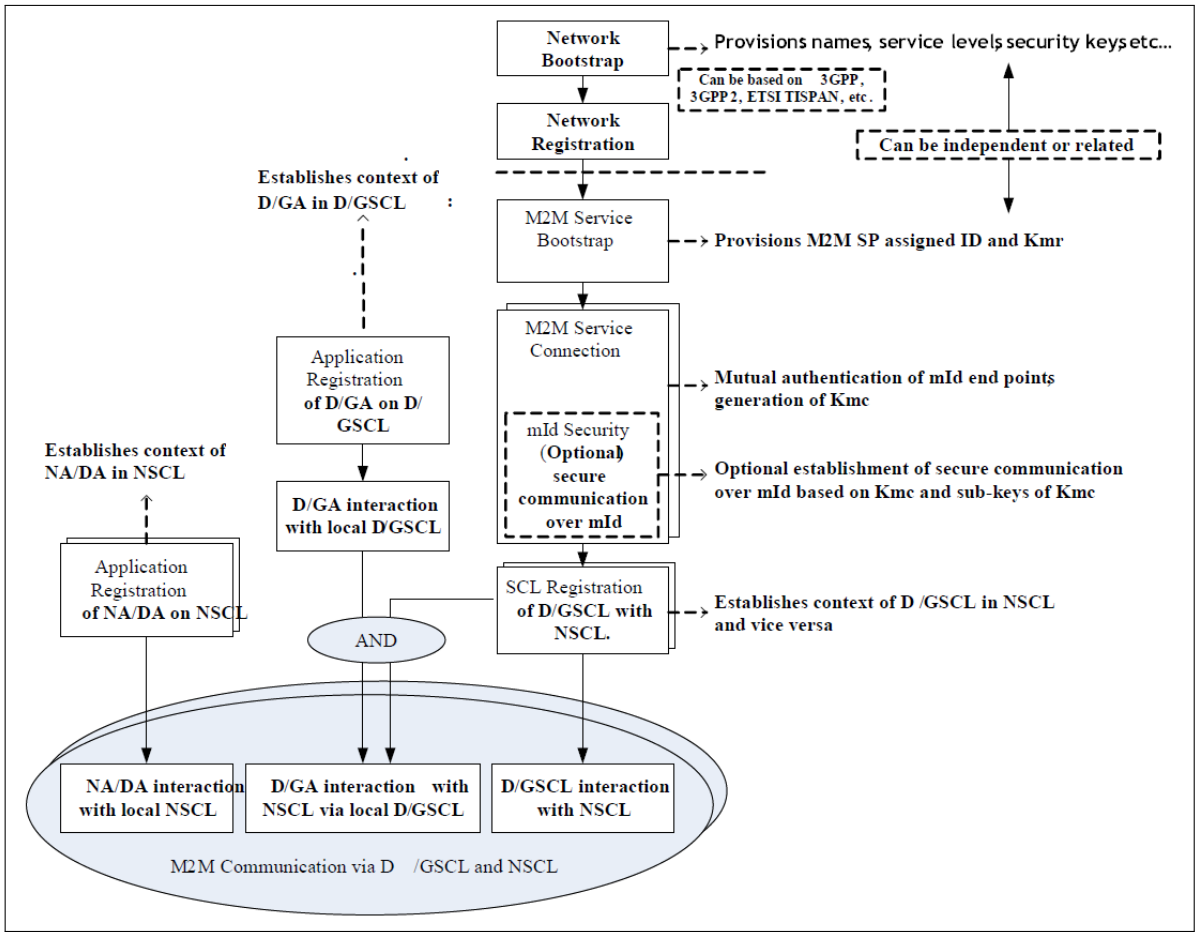

Figura 2.3: Fluxo de eventos da arquitetura ETSI padrão para comunicações M2M, retirado de [2]

numa rede M2M. Antes que tal possa acontecer, é necessário que estes tenham que ter armazenados meta-dados como nomes e chaves de segurança da rede inerente à rede M2M (a rede pode ser baseada em 3rd Generation Partnership Project (3GPP), 3GPP2, entre outras). Só após este provisionamento inicial é que o dispositivo acede à rede de suporte e seguidamente à rede M2M.

É necessário que os dispositivos/*gateways* M2M possam comunicar e sejam conhecidos pela rede M2M. Para tal tem de ser efetuado, tal como antes, um provisionamento de credenciais de segurança relacionadas coma a rede M2M. Como foi dito, estas credenciais são principalmente utilizadas para efetuar autenticação mútua nas comunicações. Assim, como se pode ver no canto superior direito da Figura 2.3, após o registo na rede (Network Registration), encontra-se um passo de provisionamento de dados M2M nos dispositivos/*gateways* M2M. Após estes últimos terem todos os dados necessários provisionados e armazenados internamente, então eles podem iniciar comunicações com a rede M2M.

O procedimento de comunicação entre o dispositivo/*gateway* M2M e a rede M2M

inclui autenticação mútua, utilizando dados previamente provisionados, assim como troca e criação de chaves para que possam ser utilizadas na cifra das comunicações que existem entre os dois componentes. É, então, estabelecida uma sessão de comunicação, opcionalmente cifrada.

O último passo para que seja possível a troca de dados relevantes entre os dispositivos/*gateways* M2M e a rede M2M é uma operação de registo. Esta operação envolve o uso de funções de capacidades de serviço M2M, referidas anteriormente, que permitem a comunicação entre as diferentes entidades M2M. Uma vez que estas capacidades estão presentes nos dispositivos/*gateways* M2M, é possível efetuar o seu registo nas capacidades de serviço presentes na rede M2M. Este registo requer uma sessão ativa, efetuada no passo anterior e tem como principal objetivo dar a conhecer o contexto e a existência dos dispositivos/*gateways* M2M à rede M2M e vice-versa. Após o registo, os componentes intervenientes podem então trocar, partilhar dados e interagir um com o outro.

O registo efetuado no passo anterior apenas permite a comunicação entre as capacidades de serviço de cada lado. As aplicações desenvolvidas, e que estão a ser executadas por ambos os componentes, ainda não podem comunicar nem interagir com outras aplicações/capacidades de serviço. Para que tal aconteça é necessário que, quer as aplicações de rede (as aplicações consumidoras de dados pertencentes na camada 6 da Figura 2.2), quer as aplicações do dispositivo/*gateways* (as aplicações produtoras que se localizam nas duas camadas mais baixas da Figura 2.2) se registem nas capacidades de serviço M2M do próprio contexto em que executam. Assim, as aplicações M2M que executam nos dispositivos/*gateways* M2M têm de se registar localmente nas capacidades de serviço M2M dos mesmos dispositivos/*gateways* M2M, de forma a que estes reconheçam o contexto e a existência das aplicações e permitam a sua interação com as capacidades de serviço da rede M2M fazendo uso das capacidades de serviço do próprio dispositivo/*gateway* em que se registaram. O mesmo acontece para as aplicações de rede: as aplicações M2M que executam na rede M2M têm de se registar localmente nas capacidades de serviço M2M da mesma rede M2M, de forma a que esta reconheça o contexto e a existência das aplicações e permita a sua interação com as capacidades de serviço da mesma rede M2M em que se registaram.

Para finalizar, para que seja possível para as aplicações M2M dos dispositivos/*gateways* M2M interagirem, através das capacidades de serviço dos últimos, com as capacidades de serviço de uma rede M2M, é necessário, obviamente, que as capacidades de serviço dos dispositivos/*gateways* M2M estejam já registadas nas capacidades de serviço da rede M2M.

A finalidade de todo este processo é fornecer uma noção do padrão de comunicação entre as máquinas que é normalmente aplicado nos sistemas IoT.

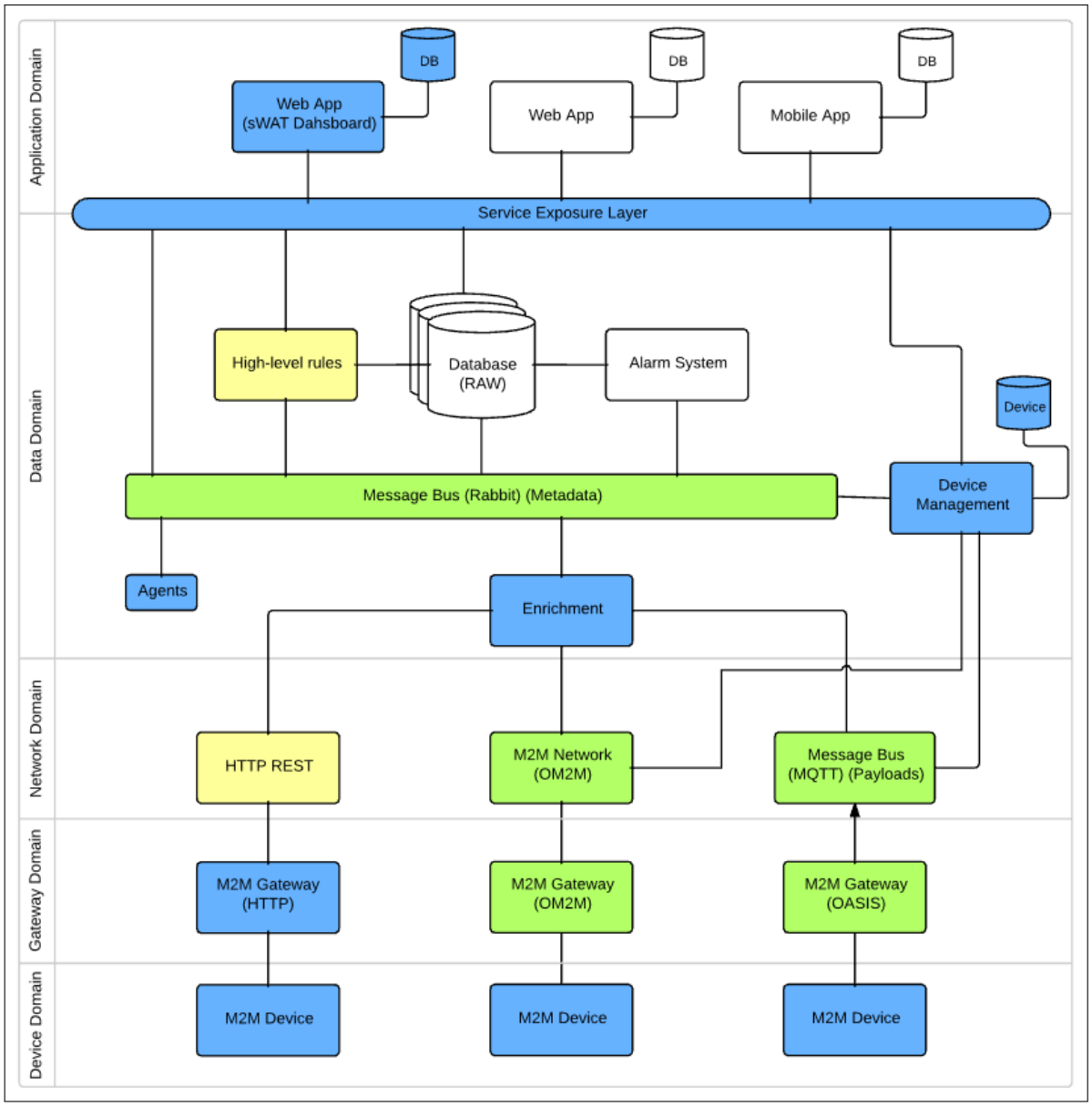

Figura 2.4: Arquitetura SCoT - plataforma tida em consideração na elaboração de uma solução segura, fornecida pelo Instituto de Telecomunicações de Aveiro

2.5 PLATAFORMA IOT UTILIZADA

Esta dissertação trata de resolver os problemas de segurança dos dados existentes numa plataforma IoT convencional. De notar que não se pretende aqui construir de raíz todos os componentes de uma plataforma IoT. De forma a prosseguir ao encontro de uma solução para os diversos problemas de segurança existentes, foi tomada em consideração uma plataforma IoT já existente: Smart Cloud of Things (SCoT).

A Figura 2.4 ilustra os componentes da arquitetura da plataforma SCoT, que não tem em consideração muitos dos problemas de segurança existentes no ambiente da IoT. Como é possível observar, nesta arquitetura existem diversos componentes já familiares.

Os dispositivos M2M estão ligados a *gateways* M2M, sendo que estes *gateways* podem implementar uma panóplia de tecnologias diferentes. Os *gateways* podem ser acedidos através de protocolos como o HyperText Transfer Protocol (HTTP); a implementação segue os padrões ETSI M2M (OM2M) e OASIS. Adjacente ao domínio dos *gateways* está o domínio de rede. Neste domínio os *gateways*, que servem HTTP, comunicam também através de HTTP REST com a plataforma IoT. No caso dos *gateways* que seguem os padrões ETSI M2M - implementação OM2M para *gateways* - comunicam com a plataforma IoT utilizando a implementação OM2M direcionada para o domínio de rede, seguindo os padrões M2M estabelecidos pela ETSI. Finalmente, os *gateways* que seguem padrões OASIS para comunicação utilizam o protocolo de mensagens Message Queue Telemetry Transport (MQTT), que funciona através da publicação e subscrição de tópicos.

No domínio dos dados existe um componente de enriquecimento que prepara os dados para serem posteriormente armazenados numa base de dados. Segue-se um barramento de mensagens. Este barramento está implementado, na arquitetura e plataforma existentes, utilizando um *broker* de mensagens: RabbitMQ. De forma semelhante ao MQTT, o RabbitMQ também foi utilizado recorrendo ao modelo de publicação/subscrição. Ligado a ele está a base de dados onde são armazenados todos os dados. A ele estão também ligados outros componentes não muito relevantes para o objetivo que é o de resolver os problemas de segurança existentes (Secção 4.1) no ambiente IoT.

No domínio aplicacional estão as aplicações Web que acedem diretamente aos dados armazenados na base de dados, passando apenas por uma camada de exposição e abstração.

Em todo este ambiente da IoT, e como será referido no Capítulo 5, foi necessário elaborar um conjunto de APIs, uma sequência de troca de mensagens com um determinado conteúdo, uma adição de novos componentes e entidades e uma modificação dos componentes já existentes de forma a transformar esta arquitetura já existente e não segura, numa arquitetura IoT segura. O foco da segurança nesta dissertação centra-se ao nível dos dados produzidos pelos dispositivos.

De forma a efetuar a validação da solução proposta, mais à frente apresentada, foi utilizado o domínio de rede desta arquitetura (uma vez que não é necessário alterá-lo para introduzir mecanismos de segurança), juntamente com o componente de enriquecimento que também não necessita de alterações, o barramento de mensagens RabbitMQ e a base de dados que apenas armazena dados em claro.

O desafio principal desta dissertação não esteve apenas em elaborar e estabelecer uma arquitetura segura no contexto do IoT. Por si só isso já provoca alguma dificuldade, tendo em conta as diversas restrições que são impostas ao nível de capacidades de

processamento, limites de comunicação com a rede, etc. No entanto, um desafio adicional foi acrescentado: a elaboração de uma arquitetura segura que proteja os dados dos utilizadores utilizando como ponto de partida uma arquitetura de uma plataforma IoT já previamente concebida. Ou seja, através de uma plataforma que já se encontra pensada houve a necessidade de lhe adicionar mecanismos de segurança dos dados tendo em conta, naturalmente, todos os componentes já existentes e a forma como estes comunicam.

Mais à frente, neste documento, na descrição da arquitetura segura proposta, será mencionado o domínio dos dados como: "Plataforma IoT". No entanto, existirão componentes adicionais que terão de ser adicionados ao domínio dos dados de forma a ser possível a implementação de mecanismos de segurança, como o leitor reparará mais à frente nesta dissertação.

De seguida os componentes da plataforma existente, e que foram utilizados na validação da solução proposta sem necessitarem de qualquer alteração ou que são componentes básicos sem complexidade no contexto da arquitetura proposta, irão ser brevemente descritos.

2.5.1 DOMÍNIO DE REDE

Este componente continua a funcionar da mesma forma nem tem intervenção direta na aplicação dos mecanismos de segurança numa plataforma IoT propostos nesta dissertação. Este componente tem o principal objetivo de armazenar informações de configuração de toda a rede IoT, guardar informações relativas à comunicação que é efetuada entre os componentes localizados em diversas máquinas (comunicações M2M), armazenar dados relativos às aplicações registadas na rede, guardar possíveis chaves criptográficas de forma a garantir segurança a níveis físicos e de comunicação ponto a ponto, armazenar eventuais dados de subscrição à rede que existem por parte de outros componentes (informações sobre a data de início e de fim da subscrição de dados num esquema de publicação/subscrição é um exemplo do tipo de dados aqui presentes), entre vários outros propósitos que, para os problemas de segurança que estão a ser tratados, não são relevantes.

Este componente recebe os dados provenientes dos *gateways* e encaminha-os, sem lhes efetuar nenhuma alteração, para o componente de enriquecimento, efetuando operações de registo das transações de dados que ocorrem no sistema. Isto acontece no caso dos dados terem como origem o dispositivo produtor passando pelo *gateway* com o objetivo de estes serem armazenados persistentemente pela plataforma IoT.

Este componente precisará de comunicar com o barramento de mensagens para que

seja possível efetuar a troca de dados entre os *gateways* aqui associados e todos os outros componentes com que pode vir a comunicar no futuro (sendo que, de momento, apenas é usado como intermediário no envio dos dados produzidos pelos dispositivos e provenientes dos *gateways*). Esta troca de mensagens terá que suportar comunicação no sentido inverso, isto é, a rede M2M (ou outro tipo de implementação como HTTP REpresentational State Transfer (REST) que não siga os padrões de comunicação M2M criados pela ETSI) terá que suportar o fluxo de dados do resto da plataforma (por via do barramento de mensagens) com destino a um *gateway*, indicado na mensagem que lhe chega, e o fluxo de dados de um *gateway* diretamente para o resto da plataforma IoT. Esta bidirecionalidade é necessária na arquitetura que é apresentada nesta dissertação. No entanto, numa plataforma convencional, pode não ser necessária dependendo das funcionalidades que a plataforma suporta.

Este componente tem, naturalmente, de ser acessível através do uso da rede pública, uma vez que os dispositivos produtores e *gateways* podem estar localizados em redes públicas e em diversos locais geográficos. Assim, este componente, apenas aceita comunicações de dados provenientes de um determinado *tenant* registado. De notar que todos estes conceitos que se irão mencionar de seguida estão associados ao modo de transmissão de dados utilizando um *gateway* que comunica com o domínio de rede através do protocolo de mensagens MQTT. Um *tenant*, nesta plataforma, é nada mais, nada menos, que um tópico MQTT que possui determinadas credenciais de uso criadas no momento de registo (nome de utilizador e *password* necessários no acesso a este tópico). Este tópico está protegido no acesso criando um ambiente isolado que pode ser utilizado. Desta forma, é possível que a rede IoT distinga diferentes tipos de serviços que estão a usar a plataforma, pois cada um poderá estar isolado no seu *tenant*, não interferindo com outros dados de outros serviços (*tenants*).

Para que seja possível que um determinado produto IoT (um produto pode ser, por exemplo, um conjunto de dispositivos utilizados para efetuar a medição de consumo elétrico em residências) possa utilizar esta plataforma é necessário que se proceda a um registo manual de um *tenant*. Dentro deste *tenant* podem estar associados/registados produtos. Dentro de cada produto estão um ou mais grupos de dispositivos e, por fim, dentro destes grupos estão os próprios dispositivos que tratarão de efetuar a comunicação com o domínio de rede (Figura 2.5). Os *tenants*, produtos, grupos e dispositivos são identificados com um Universally Unique IDentifier (UUID) [46] e estão a si associados as respetivas credenciais de acesso ao seu tópico MQTT. De notar que todo este mecanismo já se encontra elaborado, implementado e a funcionar.

Na implementação dos mecanismos de segurança de dados propostos nesta dissertação, foi usado o protocolo MQTT nas comunicações entre os produtores e a plataforma.

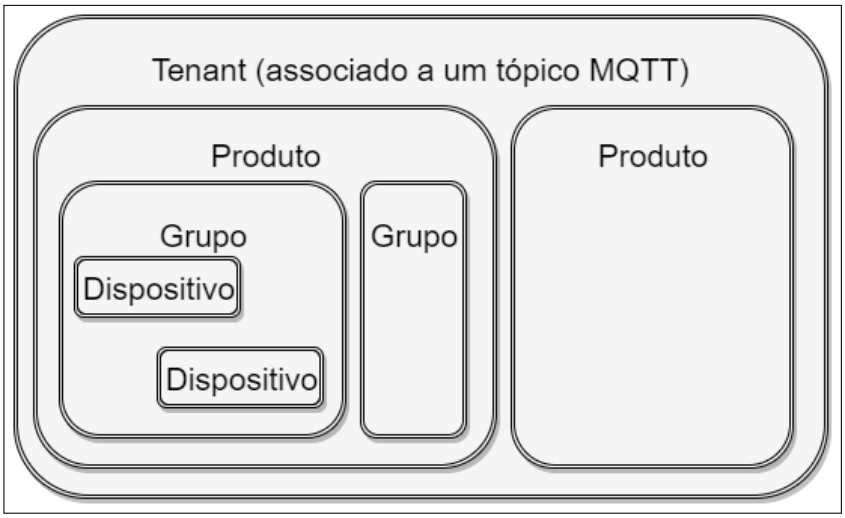

Figura 2.5: Diagrama ilustrativo da estrutura dos tenants *já existentes na plataforma IoT utilizada*

2.5.2 ENRIQUECIMENTO

Este componente não é alterado, continua a funcionar da mesma forma nem tem intervenção direta na aplicação dos mecanismos de segurança para uma plataforma IoT propostos nesta dissertação. Este componente tem o principal objetivo de enriquecer os dados que chegam a partir dos *gateways*. Os mesmos não sofreram alterações no componente da rede M2M. São efetuadas transformações e são adicionados diversos meta-dados para que os dados possam ser armazenados de forma fidedigna e para que seja possível, futuramente, conseguir recuperar novamente os dados. Após estas modificações/adições, os dados seguem para uma base de dados ou qualquer outra forma de armazenamento, passando antes por um barramento de mensagens. Na arquitetura que se propõe, a função deste componente é exatamente a mesma, sendo que o enriquecimento dos dados produzidos originalmente que chegam é feito sem este componente ter acesso ao seu conteúdo, uma vez que estes já aqui chegam cifrados e protegidos contra acesso e manipulação indevida. De notar que nem toda a informação que chega a este componente está cifrada e, por isso, é possível proceder ao enriquecimento dos dados que pode depender do domínio de aplicação dos mesmos. A adição de meta-dados não requer que este componente tenha acesso ao conteúdo dos dados recebidos, pois os meta-dados adicionados contém informação acerca do *gateway*/dispositivo produtor que está a enviar os dados (conhecido através do tópico MQTT), informação de instantes de tempo, entre outros.

Deste modo, no que toca aos mecanismos de segurança, pode-se assumir que este componente apenas redireciona os dados que lhe chegam para uma fila de mensagens.

2.5.3 BARRAMENTO DE MENSAGENS

Este componente não sofre alterações. É implementado com um (*broker*) de mensagens constituído por diversas filas de mensagens. Cada fila de mensagens está associada a um determinado destino. Ou seja, para enviar uma mensagem para um determinado componente da plataforma IoT, é apenas necessário enviar a mensagem em causa para o *broker* de mensagens com a identificação do destino onde se pretende que esta mensagem chegue. As mensagens são publicadas, pelo componente origem, na fila de mensagens apropriada de acordo com o destinatário da mensagem, o *broker* irá tratar de enviar os dados presentes nas filas de mensagens (tópicos do RabbitMQ) para os seus destinatários através de eventos. Ou seja, os dados vão sendo "despejados" das filas de mensagens e o componente da plataforma IoT em questão trata de as enviar, proativamente, para os respetivos destinatários, que estarão, evidentemente, subscritos às filas de mensagens que foram criadas para este mesmo propósito. Este "despejo" acontece até que a fila esvazie, não sendo necessário nenhum pedido de receção de dados feito por parte do destinatário resultando numa comunicação assíncrona de mensagens. A arquitetura segura proposta suporta que outros modelos de comunicação possam ser utilizados, como por exemplo o modelo de pedido-resposta. Neste modelo as mensagens poderiam ser enviadas, pelo componente origem, para a fila de mensagens apropriada de acordo com o destinatário da mensagem. Estas mensagens não são entregues de imediato. O *broker* está sempre atento a eventuais pedidos de receção de mensagens. Caso este receba um, só então é que o *broker* procede ao envio das mensagens presentes na fila de mensagem, como uma resposta ao pedido. Ou seja, os dados poderiam ser requisitados pelos destinatários, à medida que estejam disponíveis para receber novas mensagens e estabelecer comunicação com o barramento de mensagens. Após o destinatário efetuar um pedido ao *broker* de mensagens, este, tipicamente, bloqueia à espera de obter uma resposta com o estado atual das filas de mensagens em questão ou com o conteúdo das mensagens lá presentes, resultando numa interação tipicamente síncrona.

Na arquitetura proposta nesta dissertação, o modo como a informação é trocada é pouco relevante, desde que seja através de canais de seguros. Isto porque o que tem mais importância de modo a conseguir manter os dados do utilizador privados é o tipo e o conteúdo dos dados que são passados entre as várias entidades e a ordem em que essas comunicações se efetuam.

O *broker* de mensagens pode ser mais complexo do que foi aqui explicado. Podem existir várias filas de mensagens para os mesmos destinatários onde cada fila pode estar associada a um determinado tipo de mensagens ou a um determinado nível de prioridade. Nesta dissertação é assumido que este componente apenas trata de encaminhar os dados a partir de uma origem para o destinatário correto recorrendo a tópicos RabbitMQ,

sem que seja efetuada qualquer alteração aos mesmos.

O acesso a este barramento de mensagens é controlado. Para ser possível aceder-lhe, é necessário introduzir um nome de utilizador e uma palavra-passe. Estas credenciais são as mesmas para todos os componentes que pretendam aceder ao barramento, no entanto, o acesso a este componente da plataforma é limitado uma vez que se encontra protegido numa rede interna e não acessível do seu exterior. Tipicamente, para efetuar o acesso a este componente por parte de componentes externos à plataforma, são utilizadas APIs. No caso deste trabalho, foi necessário proceder à implementação de algumas APIs que permitissem o acesso a este barramento através de diversas operações e passagem de parâmetros controlado com origem externa à plataforma.

2.5.4 BASE DE DADOS

Este componente mantém-se igual à sua implementação numa plataforma IoT convencional. Na arquitetura IoT onde serão implementados os mecanismos de segurança, este componente é constituído por uma base de dados distribuída. No entanto, não teria que ser assim, o componente de armazenamento de dados pode ser um serviço externo como a *cloud* ou um sistema de ficheiros ou base de dados interna. Este componente, pode, alternativamente, pertencer à plataforma e ter como função comunicar com um determinado serviço externo que proceda ao armazenamento dos dados. No fundo, não importa quais os mecanismos utilizados para o armazenamento dos dados. Existe apenas um requisito, que é o de este componente possuir uma interface que permita adicionar novos dados para armazenamento persistente e poder recuperar os dados anteriormente armazenados. Uma vez que, normalmente, estes componentes possuem mecanismos de autorização, o barramento de mensagens tem de ser autorizado a enviar os dados cifrados dos produtores para serem armazenados aqui e os componentes que tratam de recuperar os dados posteriormente também têm de ser autorizados para tal. Relativamente à autorização do barramento de mensagens para armazenamento de dados, é necessário que existam mecanismos que não permitam o armazenamento de dados que outros componentes ou entidades geram. Para tal, o acesso à base de dados é feito utilizando uma chave de API. Esta chave é fornecida no momento de criação de um *tenant*. O seu criador poderá utilizá-la para, de alguma forma, determinados consumidores poderem aceder aos dados. A chave de API está associada ao *tenant* no momento de criação.

De relembrar que os dados que chegam a este componente podem ou não estar cifrados (armazenamento de dados cifrados que será introduzido pela arquitetura proposta nesta dissertação). Assim, caso seja utilizado um serviço externo para o armazenamento,

não existe perigo no que toca à confidencialidade dos dados lá guardados, desde que a arquitetura proposta esteja a ser utilizada. Como também é adicionado um mecanismo de manutenção da integridade como será explicado na Secção 5.5, caso estes sejam manipulados este facto será mais tarde detetado.

2.5.5 CAMADA DE EXPOSIÇÃO DE SERVIÇOS (APIS)

Este componente tem a principal funcionalidade de fornecer interfaces às aplicações consumidoras desenvolvidas. Esta interface expõe as diversas operações que as aplicações podem efetuar sobre a plataforma IoT. Existem vários casos de uso de uma plataforma IoT (Secção 2.2) e, por isso, este componente pode depender de caso para caso em que é aplicado. Por exemplo, no caso de uso da domótica na IoT, este componente necessitará de expor certas funções de controlo e acesso a funções presentes nos diversos dispositivos "produtores". Por exemplo, a API deve incluir funções que permitam a comunicação com um determinado dispositivo selecionado para que seja possível ligá-lo, desligá-lo ou ordenar que certas operações sejam nele efetuadas.

No caso de uso da plataforma IoT onde os dispositivos produtores são contadores de eletricidade inteligentes e as aplicações consumidoras apenas necessitam de obter os dados produzidos pelos medidores, esta API necessitará de permitir essa recuperação, assim como um eventual registo nesta plataforma.

2.5.6 APLICAÇÕES CONSUMIDORAS

As aplicações consumidoras estão associadas a uma plataforma IoT e são, tipicamente, pertencentes a empresas que necessitem de efetuar um acesso a dados produzidos para executar, por exemplo, um estudo sobre como os seus clientes utilizam os dados. Estas aplicações vêm a plataforma como uma origem de dados, ou seja, após todos os mecanismos de registo terem sido concluídos, a aplicação, de forma a obter os dados pretendidos, procede a um pedido pelos mesmos. Este pedido irá resultar, mais tarde no retorno dos dados dos seus clientes. Como veremos mais à frente, o controlo de acesso e manutenção de privacidade destes dados é o foco central desta dissertação, pelo que, o pedido de dados à plataforma, irá ser um pedido com alguma informação adicional relativamente à arquitetura inerente à plataforma IoT considerada como origem. No entanto, continua a ser a própria plataforma a tratar de toda a complexidade que a operação de recuperação de dados possa acarretar.

As aplicações consumidoras terão de ser alteradas e implementadas de acordo com a arquitetura que é, nesta dissertação, apresentada.

Desta forma, este capítulo é concluído. Foram apresentados os componentes base da plataforma IoT utilizada indicando as principais funções dos componentes já existentes.

capítulo 3

Estado da Arte

Neste capítulo são descritos os trabalhos estudados, analisados e avaliados.
São mencionados os pontos fracos de cada um dos trabalhos de investigação já feitos na mesma área ou em áreas adjacentes à que está a ser tratada nesta dissertação.

Neste capítulo são descritos e avaliados trabalhos de investigação relacionados com a temática desta dissertação. Estes trabalhos procuram criar soluções de segurança sobre os dados privados de utilizadores ou que procuram analisar e listar os problemas atuais existentes nas plataformas IoT com especial foco na manutenção da privacidade do utilizador, uma vez que os dados são enviados para uma plataforma IoT terceira.

Existe pouco trabalho executado na área da segurança ao nível dos dados. As razões para a existência deste facto podem ser várias:

- Maior preocupação em construir uma plataforma IoT que seja, acima de tudo o resto, funcional;
- Complexidade da conceção de uma arquitetura de segurança que resolva uma grande quantidade de problemas de segurança existentes;
- Dificuldade em aproveitar e aplicar os métodos de segurança utilizados, já existentes e que resolvem a maioria dos problemas de segurança na Internet ao cenário da IoT e a plataformas IoT já existentes para que se satisfaçam as necessidades de segurança dos dados;
- A necessidade de separar poderes por diferentes entidades não é facilmente conseguida.

Como será referido detalhadamente nesta secção, existe uma grande quantidade de trabalhos, notícias e artigos sobre a temática desta dissertação que apenas enumera e explica os problemas de segurança dos dados em plataformas IoT que existem atualmente. Ou seja, não propõem soluções para a resolução desses problemas, limitando-se apenas

a mencionar quais são os problemas que devem ser resolvidos para que uma plataforma IoT possa ser segura.

Outra parte do trabalhos que foram analisados estudaram novos mecanismos de cifra e de controlo de acesso que podem ser utilizados em alternativa aos métodos de segurança atualmente existentes e que foram usados na Internet. Estes métodos, não tendo sido suficientemente expostos a testes de vulnerabilidades em grande escala, podem não ser a melhor escolha para serem implementados em plataformas reais. Este fator é extremamente importante na área da segurança. Os métodos de segurança atualmente mais conhecidos como as funções de dispersão criptográficas Secure Hash Algorithm (SHA) [5] e Message-Digest algorithm 5 (MD5) [66] ou os algoritmos de cifra por blocos Data Encryption Standard (DES) [52] e Advanced Encryption Standard (AES) [15] foram submetidos a inúmeros testes de vulnerabilidade por parte de terceiros. Foi demonstrado que alguns destes métodos, como a função *hash* MD5 e o algoritmo de cifra por blocos DES, eram vulneráveis e caíram em desuso ao longo do tempo. É, assim, um risco elevado se forem utilizados novos métodos de segurança num sistema real sem que tenham sido submetidos primeiramente a testes de vulnerabilidade em grande escala. A maioria dos novos métodos estudados e apresentados nos trabalhos oferecem algumas vantagens mas pecam na falta de flexibilidade de utilização dos dados como é pretendido na IoT e/ou na falta de segurança a outros níveis.

Outros trabalhos apresentam uma arquitetura a ser incluída/respeitada numa plataforma IoT para que seja possível atingir o principal objetivo de segurança definido pelos respetivos autores. No entanto, as soluções apresentadas apenas resolvem uma parte daquilo que são os principais problemas de segurança existentes sobre os dados. Nestas soluções não estão, por isso, incluídas arquiteturas globais que resolvam os vários problemas de partilha de dados privados com terceiros no contexto da IoT sem expor os dados a vulnerabilidades ou tornando o uso dos mesmos inflexível, como é analisado nas subsecções seguintes. Muitas destas soluções nunca poderiam ser aplicadas à IoT pelas mais diversas razões. Por exemplo, alguns contemplam o assunto da faturação de acordo com os dados (valores) que os dispositivos de medidas registavam. Nesta dissertação não serão abordados sistemas de faturação, focando-nos mais na proteção da análise dos dados de dispositivos criando uma vista geral sobre esses mesmos dados.

3.1 ANÁLISE DAS NECESSIDADES DE SEGURANÇA

No trabalho [70], o autor menciona a falta de privacidade que atualmente os utilizadores/produtores de dados de uma plataforma IoT têm relativamente aos seus dados. Os produtores de dados deixam de ter controlo sobre os seus próprios dados.

Estes, juntamente com meta-informação acerca do seu criador, são enviados para fora de um dispositivo e saem do controlo do dono dos dados para sempre. O utilizador/dono dos dados fica sem saber onde é que toda essa informação (privada) se encontra na Internet e por quem é acedida, não existindo qualquer controlo do dono no acesso aos seus dados, na maioria das vezes sensíveis e privados.

O autor destaca 4 requisitos gerais para que se possa obter segurança e privacidade em plataformas IoT. São estes:

- Resiliência a ataques: O sistema tem de evitar pontos únicos de falha e ajustar-se a eventuais falhas em nós;
- Autenticação de dados: A informação dos identificadores e dos objetos recuperados têm de ser autenticados;
- Controlo de acesso: Os fornecedores da informação têm de ser capazes de implementar controlo de acesso a informação requerida;
- Privacidade do utilizador: Têm de ser tomadas medidas para que apenas o fornecedor de informação seja capaz de inferir algo através da utilização do sistema de pesquisa sobre um utilizador específico.

Em [70] também se listam diversas tecnologias de reforço de privacidade que já existem e que podem ser utilizadas. Uma das tecnologias são as Virtual Private Networks (VPNs). No entanto, é mais frequentemente usada em grupos fechados não existindo qualquer privacidade dentro do mesmo grupo. Esta solução não permite troca de informação dinâmica e global e não é prática para terceiros fora da rede. Outra tecnologia é o Transport Layer Security (TLS). Esta tecnologia assegura a confidencialidade, integridade e autenticação nas comunicações. [70] ainda mencionou outras tecnologias que garantem autenticação como o Domain Name System Security Extensions (DNSSEC) [7], o *Onion Routing* [53] e o Private Information Retrieval (PIR) [12], sendo que estas tecnologias estão mais relacionadas com problemas de segurança na transmissão de baixo nível dos dados e no *lookup* de nós e tradução de nomes, enquanto que nesta dissertação existe uma maior preocupação na existência de privacidade a alto nível (acesso aos dados por terceiros) na transferência de dados entre a camada dos dispositivos/*gateways* e a camada terceira que é a plataforma IoT e/ou a *cloud*. A forma como os dados podem ser acedidos por entidades terceiras também é abordada nesta dissertação.

Ainda em [70], na restante parte, fala-se sobre questões legais e sobre a lei que é necessário respeitar no que toca à privacidade e controlo de dados sensíveis.

Um livro branco da Wind River [4], empresa líder de *software* móvel para dispositivos embutidos, aborda os problemas e as necessidades de segurança que uma plataforma IoT necessita de alcançar. Os autores referem, e concordamos, que não existe nenhuma "bala de prata" (*silver bullet*) para mitigar todos os ciberataques possíveis. Ainda

mencionam que o que se deve fazer para construir uma plataforma segura é aplicar e adaptar os mecanismos de segurança já existentes à IoT, pois estes são largamente usados e já foram extensivamente testados. No entanto, existe o problema de estarmos a lidar com dispositivos de natureza computacional fraca que foram desenhados para um determinado propósito e não contemplando os problemas de segurança de uma plataforma. Este facto é uma grande restrição, uma vez que os métodos de segurança existentes podem ser computacionalmente intensivos e a sua aplicação em dispositivos condicionados poderia afetar em muito o desempenho geral da plataforma ou até mesmo levar ao bloqueio dos dispositivos, o que não é desejável, principalmente quando se trata de equipamentos que transmitem dados em tempo real.

A Wind River lista e descreve os novos perigos, restrições e desafios de segurança que surgem com a chegada e aplicação da IoT. Como foi mencionado acima, um dos problemas é a existência de dispositivos com pouca capacidade a nível computacional. Os dispositivos que agregam a informação (os produtores) são dispositivos básicos onde não é possível implementar mecanismos que requeiram, por exemplo, muito espaço em disco ou uma grande capacidade de processamento. Estes dispositivos são, normalmente, desenhados para baixo consumo e possuem ligações externas limitadas. Tipicamente têm a memória e capacidade de processamento suficientes para os seus fins e são autónomos, uma vez que não requerem que um ser humano lhes aceda para o operar nem requerem credenciais de acesso. Têm de efetuar os seus próprios juizos e decisões.

São levantadas algumas questões acerca de como manter os dados que saem, por exemplo, de um contador de energia privados para que não sejam acedidos por terceiros sem a respetiva autorização; ou como é que os Controlador Lógico Programáveis (CLPs) que operam sistemas robóticos se podem proteger da interferência humana; ou como pode um sistema de controlo de reatores nucleares obter atualizações de *software* e *patches* de segurança.

A Wind River, por fim, apresenta uma série de mecanismos que se devem aplicar nos dispositivos por forma a garantir os principais requisitos de segurança. Estes mecanismos focam-se demasiado na segurança do dispositivo agregador de informação e não tanto na segurança e privacidade dos dados que transmite para fora do seu contexto, pelo que não descreveremos pormenorizadamente os diferentes mecanismos. Estes são 5: *secure booting* - avaliar a validade do software que vai executar através da verificação de assinaturas; controlo de acesso - limitar o acesso por parte de aplicações e componentes do dispositivo às diferentes partes do sistema operativo; autenticação - antes da transmissão de dados para a rede, é necessária a autenticação das entidades que estão a comunicar; *firewalling* - é necessária a existência de *firewalls* para a filtragem e inspeção de pacotes; e por fim atualizações e *patches* - manter os dispositivos sempre atualizados principalmente com correções ao nível da segurança.

Em [57] são listados e descritos os problemas de segurança que existem atualmente na maioria das arquiteturas IoT existentes.

Os autores começam por definir a IoT e estabelecer dois princípios inerentes à IoT que necessitam de serem tidos em conta no momento da elaboração de uma arquitetura para uma plataforma. Os princípios são: (1) a localização do conhecimento, informação e/ou dados encontram-se na borda da rede assim como o aprovisionamento de serviços e aplicações; (2) existe uma extensa colaboração entre diversas entidades e/ou componentes da rede. Estes princípios são característicos da IoT. Uma plataforma IoT possui certos requisitos e propriedades que necessitam de ser tratados, de forma a oferecer vantagens na utilização de uma plataforma IoT relativamente a uma plataforma e sistema convencional de partilha e agregação de dados provenientes de sensores. Essas propriedades são: *abertura*: é necessário que uma plataforma IoT ofereça meios de inclusão de aplicações e serviços complexos desenvolvidos por terceiros através de uma API. Sem este requisito/propriedade, não é possível aproveitar as vantagens de uma plataforma deste género. A plataforma tornar-se-ia de uso estático e/ou privado; *viabilidade*: esta propriedade engloba dois aspetos: o quão viável é a tecnologia para que ela possa entrar no mercado e se uma empresa está disposta a correr o risco de depender de um fornecedor por um longo período de tempo; *confiança*: a plataforma IoT necessita de ter uma gande disponibilidade assim como um bom desempenho; *escalabilidade*: é necessário que a plataforma possa suportar a quantidade crescente de dispositivos, aplicações e dados que se vão adicionando no futuro; *interoperabilidade*: os componentes heterogéneos devem ser capazes de interagir entre si; *problemas de segurança*: é fundamental que os problemas de segurança nas comunicações e a privacidade dos dados dos utilizadores seja conseguida para que a plataforma IoT não falhe e seja confiável.

Os autores identificaram diversas falhas de segurança presentes em arquiteturas IoT que necessitam de ser resolvidos.

O primeiro problema de segurança numa plataforma IoT é o problema de identificação e autenticação. Como garantir que um determinado componente da plataforma está a comunicar com o componente "correto" e não com um impostor? Para que a plataforma e os seus serviços inerentes sejam confiáveis e seguros contra ataques de falsa identidade é necessário aplicar mecanismos de identificação e autenticação em todos os componentes. Como proceder à identificação da grande quantidade de dispositivos e componentes que existem na IoT é um problema que é necessário solucionar. O facto das interações serem dinâmicas, isto é, existem múltiplos componentes que podem comunicar com múltiplos componentes numa plataforma, dificulta ainda mais este processo. Os autores sugerem, como uma possível solução, a existência de um componente centralizado que regista, identifica e autentica cada um dos diferentes componentes, no entanto, possui algumas desvantagens como sendo um ponto único de falha e a necessidade de efetuar muitas

comunicações com o componente de autenticação, o que poderá diminuir o desempenho da plataforma.

Outro problema de segurança está no controlo de acesso. Como filtrar os acessos das várias aplicações/serviços (consumidores), que se encontram na borda da rede, aos diferentes dados recolhidos pelos produtores (dispositivos/sensores que agregam os dados)? Não se pode fornecer a mesma informação a todos os requisitantes, pois nem todos têm o mesmo direito (de acordo com as suas qualificações) de acesso a certos dados armazenados pela plataforma. Para tal, os autores sugerem a implementação de um controlo de acesso também centralizado utilizando estruturas como Access Control Lists (ACLs) e Role-Based Access Control (RBAC) [23]. No entanto estruturas como o RBAC necessitam de ser adaptadas uma vez que podem existir diferentes contextos para os diferentes papéis que seriam definidos. Outra sugestão é que a lógica de controlo de acesso seja passada a uma outra entidade confiável utilizando *tokens* de acesso, como é o caso do Kerberos [49].

Um outro problema está na segurança de protocolos e nas comunicações de rede. Os autores sugerem o uso de canais seguros TLS [19] ou Datagram TLS (DTLS) [54] sobre todas as comunicações utilizando certificados X.509 [35].

Um outro problema, e aquele em que nos iremos centrar nesta dissertação, é a privacidade. Como controlar os dados produzidos pelos dispositivos/sensores (produtores) para que terceiros não lhes possam ter acesso? Os autores fazem algumas sugestões ao nível do dispositivo, como desenvolver os dispositivos de modo a controlarem eles mesmos que tipo de informação é que deve ser enviada para terceiros ou a controlarem a granularidade da informação que enviam: um exemplo é a comunicação de localizações GPS em que se podem enviar dados sobre a localização em forma de área e não a localização exata do dispositivo. No entanto, esta solução não seria muito flexível para as aplicações que utilizam os dados, tornando-as menos precisas. Os dispositivos poderiam também, segundo os autores, controlar o acesso aos dados produzidos pelo mesmo através de mecanismos de controlo de acesso. Contudo, esta forma de proteção não é recomendada, o que implicava que o dispositivo tivesse uma capacidade de armazenamento grande para que os dados de controlo de acesso pudessem ser armazenados. Além das capacidades de processamento e de memória reduzidas destes equipamentos, os mesmos têm conexão limitada. Implementar controlo de acesso ao nível do dispositivo iria ser prejudicial para o desempenho do mesmo, uma vez que este teria que tratar muitas ligações de rede ao mesmo tempo. Os mesmos autores referem ainda como uma solução o uso de cifras homomórficas, entraremos em mais detalhes sobre este tipo de cifras na Secção 3.2.2. Ainda no domínio dos problemas de privacidade dos dados, existem ainda outras situações que necessitam de ser tidas em conta. O dispositivo produtor que envia os dados para terceiros necessita de ser confiável pelo dono do

mesmo. Este dispositivo pode conter código malicioso que localiza e segue o dono (utilizador) sem o seu consentimento ou partilha informações com outras entidades que não a plataforma na qual está inserido. Os autores sugerem um ambiente de sistema operativo controlado onde os dados de entrada e saída são controlados e o sistema é examinado frequentemente para deteção de *software* malicioso. Estes últimos casos não serão tratados nesta dissertação e será assumido que os dispositivos funcionam corretamente e como esperado no envio dos seus dados.

Um outro problema de segurança encontra-se na relação de confiança que existe nos diversos componentes. Esta relação pode nem sempre verificar-se, uma vez que qualquer componente terceiro, que à partida é totalmente confiável, pode ser alterado e pode transferir dados e informação duvidosa para as outras entidades. Para tal os autores propõem um sistema de classificação onde cada entidade da rede possui uma tabela com a "quantidade" de confiança que deposita sobre outras entidades. Um exemplo está na confiança que os consumidores depositam sobre os dispositivos produtores. Para tal, seria necessário detetar certas anomalias no conteúdo dos dados que lhes são chegados para que fosse possível detetar irregularidades e, consequentemente, baixar a reputação desse dispositivo. Os autores propõem ainda que as diferentes entidades comuniquem de forma existir uma partilha de reputação.

Por fim, os autores afirmam que existe o problema da tolerância a falhas, que não será aqui detalhado.

[28], [48] são dois trabalhos que efetuaram uma análise sobre o uso de dados de contadores/medidores inteligentes de eletricidade de uma casa para conseguir obter informações sensíveis acerca do que se está a passar dentro da mesma.

Em [48], o autor analisa o quão importante é a privacidade dos dados que os contadores inteligentes agregam e enviam para fora. Estes dispositivos liberam informação de consumo de energia associada a um determinado momento temporal que pode ser usada por terceiros para inferir com grande precisão aquilo que está acontecer dentro da propriedade privada onde o contador está a atuar. Esta informação pode ser bastante útil para um atacante, pois este pode, por exemplo, obter informações preciosas sobre a atividade que existe dentro de casa associada a um determinado intervalo de tempo. Ao conseguir esta informação, um atacante pode aperceber-se que não existe ninguém em casa entre as 10 horas e as 12 horas da manhã durante a semana (informação esta que provém da análise de utilização de eletricidade a essa hora) e pode tentar planear um assalto a essa residência e tornar o seu ataque malicioso num sucesso com uma elevada probabilidade.

Pode ainda ser possível obter outras informações acerca da residência, para além da inatividade num determinado intervalo de tempo. Um exemplo é a determinação se existe um recém-nascido dentro de casa. Para tal, o atacante poderia analisar se,

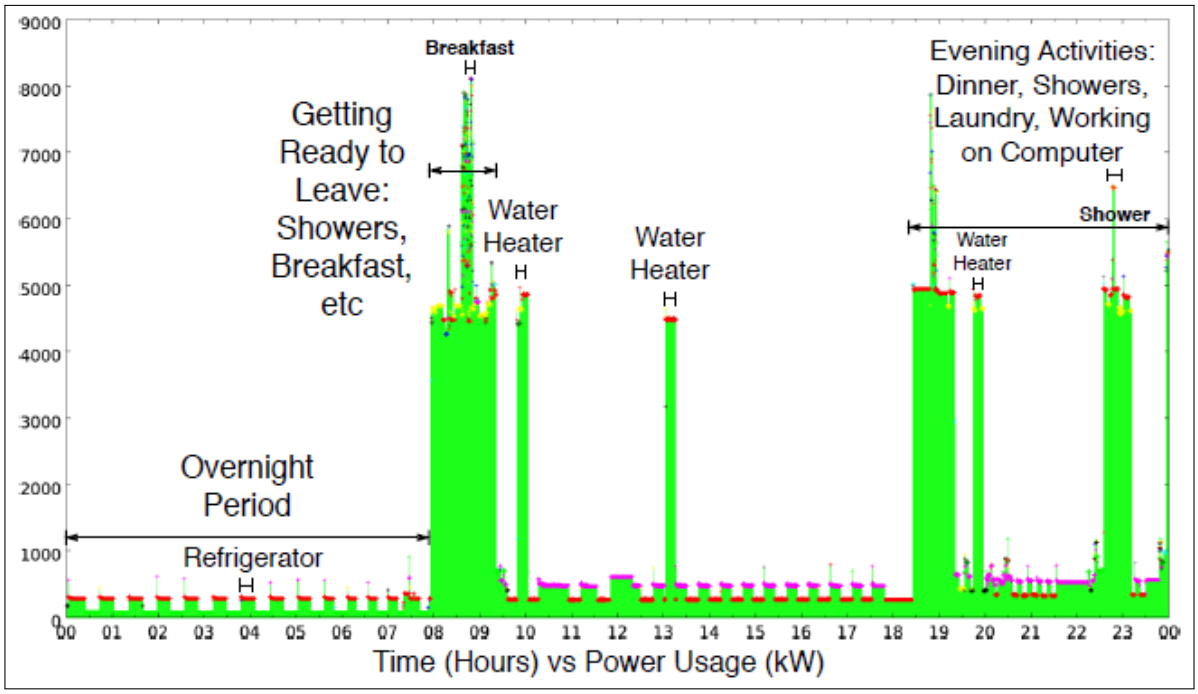

Figura 3.1: Resultado da análise dos dados de um contador inteligente feito por [48]

durante a noite, existe consumo de energia e de quanto em quanto tempo. Se tal acontecer, é provável que exista um recém-nascido dentro de casa e este é regularmente alimentado durante a noite.

Os autores, através do seu trabalho, demonstram como se pode obter tal informação acerca dos utilizadores de contadores inteligentes. Os autores utilizaram uma granularidade de 1 segundo nos seus contadores. Os principais passos para a obtenção desta informação são: (1) pré-processamento dos dados adquiridos pelo contador usando um algoritmo de *clustering* já existente de forma a agrupar usos de energia semelhantes; (2) etiquetar cada evento de energia com uma ou mais características que o distingue; (3) filtrar as aplicações automáticas que consomem energia analisando os momentos de baixa atividade e (4) mapear as etiquetas a eventos reais. O resultado final destes passos é um gráfico como o apresentado na Figura 3.1. É possível verificar nesta imagem que, durante a noite, a energia consumida é de apenas um frigorífico. Esta conclusão é feita através de conhecimentos externos acerca dos padrões de consumo de energia que são característicos dos diferentes eletrodomésticos. Entre as 8 horas e as 9 horas e 30 minutos, existe uma grande quantidade de energia a ser consumida. Presume-se ser da atividade das pessoas da casa depois de se levantarem e antes de saírem para o trabalho. Como se pode verificar, existem picos de energia. Estes picos foram analisados pelos autores que chegaram à conclusão serem provenientes de um aquecedor de água. Finalmente, verifica-se que existe muita atividade de consumo energético dentro de casa pela hora do jantar. Este tipo de análise é um risco para os habitantes desta

residência, uma vez que o seu quotidiano dentro de casa é revelado para terceiros. Os quais podem-se aproveitar da situação para planear roubos ou assaltos à casa em horas de pouca atividade, por exemplo.

De seguida os autores apresentam uma arquitetura de solução para o problema da privacidade dos dados. Esta arquitetura consiste na comunicação de diferentes contadores inteligentes de diferentes residências com um *gateway* da vizinhança. É através deste *gateway* que os dados passam para os consumidores dos mesmos, no entanto, estes dados não têm qualquer informação sobre quem os originou e, antes de serem enviados para os consumidores, é aplicada uma função de soma onde o total de potencial energético medido é, então, comunicado. Esta informação esconde em demasia os dados reais, não se podendo obter informações e fazer estudos precisos sobre os dados dos diferentes dispositivos. Esta função aplicada no *gateway* deveria poder ser uma qualquer definida pelo consumidor dos dados, não tornando a análise dos dados tão granular quanto possível e pouco flexível.

Os autores desenvolveram esta arquitetura com o principal objetivo de conseguir efetuar leituras para efeitos de faturação. Para tal, os autores utilizaram protocolos Zero-Knowledge (ZK) [22] que são resilientes a manipulações de dados (por parte dos clientes). Através da informação disponibilizada pelo *gateway* e com o uso de protocolos ZK é possível comunicar diretamente com os dispositivos medidores, utilizando várias rondas de *challenge-response*, para obter informação real sobre o total a ser pago na respetiva residência sem que sejam revelados todos os dados adquiridos pelo contador. No entanto, os autores não tiveram em conta a grande limitação e baixa capacidade de processamento dos diversos dispositivos. Uma inclusão de procedimentos criptográficos nos próprios dispositivos poderia levar ao bloqueio dos mesmos e iriam afetar o desempenho da plataforma.

Em [28] o autor tem um objetivo semelhante ao trabalho anterior sem a presença de uma arquitetura solução. Os autores demonstraram que é possível, através da análise dos dados enviados por um contador inteligente de eletricidade, obter informação sobre o canal de televisão em que uma televisão está ligada. Para além disso, é possível também identificar conteúdo audiovisual (protegido por *copyright*) que está atualmente a ser visualizado num televisor CRT, num plasma ou num LCD. Os autores desenvolveram uma função de previsão de energia consumida, analisando e calculando os gastos de energia que uma determinada imagem de um filme, por exemplo, necessitava. Uma vez que as televisões utilizam o modelo de cores RGB, é possível calcular a quantidade de cor que um determinado pixel necessita e, portanto, calcular a energia consumida por um *frame* do vídeo (filme). Ao efetuar esta análise para todos os *frames* do filme, é possível construir um gráfico que modele, ao longo do tempo do filme, a quantidade de energia necessária pelo televisor para construir uma a imagem no ecrã. Comparando

estes dados com os dados lidos de um contador inteligente (que monitorize apenas a energia consumida de uma televisão), é possível obter informação sobre os conteúdos que estão a ser visualizados.

Acrescentando a este estudo sobre as necessidades de segurança que é necessário empregar para conseguir construir uma arquitetura de uma plataforma de IoT segura, foram analisados dois artigos apresentados em dois *websites* diferentes [20], [30]. Ambos referem o facto de cada vez mais *startups* estarem a iniciar o seu negócio na área da IoT, no entanto, a maioria foca-se na funcionalidade da plataforma que desenvolvem sem terem em conta os problemas de segurança que existem, principalmente ao nível da privacidade e acesso aos dados. Um dos artigos refere a necessidade da inclusão de mecanismos de segurança no modelo de arquitetura de uma plataforma IoT a ser desenvolvida logo a partir da fase inicial. Isto é, antes do começo de qualquer desenvolvimento, é preciso ter em conta a segurança a partir do início do planeamento da arquitetura. Para além desta metodologia, é necessário saber quem é o nosso inimigo, como é que ele pode atuar sobre aquilo que se possui (ou que se irá desenvolver), possíveis pontos e tipos de ataque sobre a arquitetura e como nos prevenir face a tais invasões por parte de atacantes.

Finalmente, é necessário que a plataforma esteja preparada para o caso da mesma ser atacada no caso de existirem brechas de segurança. Por fim, e para terminar esta secção, foi analisado um documento [51] que identifica os principais problemas/desafios existentes no que toca à privacidade dos dados. O primeiro problema é a ubiquidade da recolha de dados sobre informações, hábitos, localizações e condições físicas pessoais. Atualmente, e com a tecnologia já existente, é praticamente impossível não deixar um rasto na rede que faz parte das nossas vidas, a Internet. A introdução de sensores e dispositivos em espaços privados, como carros, casas e o próprio corpo das pessoas, introduz desafios de privacidade. Estes dispositivos ligados enviam dados privados para as empresas que deles "necessitam", porém não é conhecida a sua utilização real e, na maior parte das vezes, assumimos que essas empresas são seguras e confiáveis, o que pode não acontecer. Um exemplo deste tipo de situações são, por exemplo: uma televisão inteligente que envia informação sobre os canais que cada utilizador visualiza (*reality shows*, canais de história, etc.) - serão estes dados partilhados com universidades ou até mesmo possíveis empregadores? Ou irá esta informação ser usada como uma imagem/perfil do utilizador da televisão que será visível para outros mas não para o próprio utilizador? Para resolver estes tipos de problemas é necessário conceptualizar e implementar uma arquitetura que não degrade as funcionalidades do IoT e que fornece tanta privacidade aos dados quanto possível.

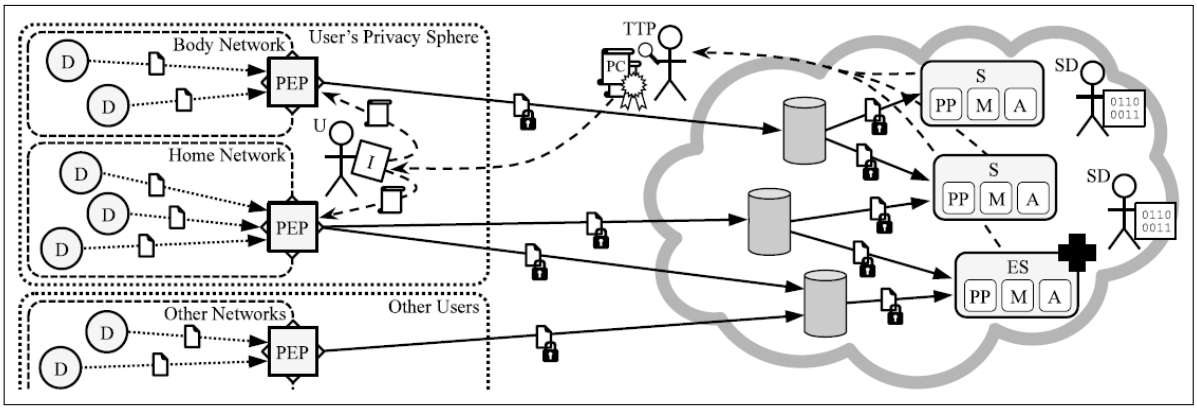

Figura 3.2: Arquitetura IoT proposta por [32]

3.2 SOLUÇÕES

3.2.1 MÉTODOS DE SEGURANÇA EXISTENTES

Nesta secção serão analisados alguns dos trabalhos que já foram desenvolvidos na área de segurança e privacidade dos dados, seu armazenamento e controlo de acesso aos mesmos.

Os autores de [32] sugeriram uma arquitetura de aplicação de privacidade centrado no utilizador (o produtor e dono dos dados) no contexto da IoT. Os autores apresentam uma arquitetura inicial básica onde são visíveis os componentes principais de uma plataforma IoT. Existem dispositivos que se ligam a *gateways* por onde passam todos os dados. Os autores pretendem criar um perímetro de privacidade dos dados à volta dos *gateways* e dos dispositivos produtores de informação. Neste perímetro encontra-se o utilizador. Os dados provenientes dos *gateways* seguem diretamente para bases de dados onde são armazenados. Os consumidores (serviços ou empresas), sempre que necessitem de aceder aos dados, procedem à execução de uma *query* às bases de dados para assim obter a informação que pretendem.

Para resolverem o problema da privacidade dos dados, os autores apresentam a arquitetura presente na Figura 3.2. As diferenças da arquitetura que contempla mecanismos de segurança dos dados relativamente à arquitetura inicial, típica de uma plataforma IoT, está na introdução de Pontos de Reforço de Privacidade (PEP) nas *gateways*, na introdução de Políticas de Privacidade (PP), Monitorização de Utilização de Dados (M) e de informação de Auditoria de informação (A) nos serviços e na introdução de uma TTP. O PEP tem a principal função de cifrar os dados antes de os enviar para a rede e só depois é que os dados são armazenados persistentemente. Esta cifra é uma cifra simétrica efetuada com chaves simétricas. Após isso, a chave simétrica é cifrada com a chave pública da *cloud* (bases de dados). Todos estes dados são enviados então para a *cloud*. O principal problema surge aqui: a *cloud* pode usar a sua chave

privada para decifrar a chave simétrica que protege os dados e, de seguida, usar esta mesma chave sobre os dados para obter os dados em claro. Assim, a privacidade dos utilizadores que produzem os dados pode ser violada pela *cloud*. Os autores centram-se também na possibilidade dos utilizadores controlarem os diferentes níveis de privacidade dos dados: os utilizadores podem definir no PEP (*gateway*), através de uma linguagem proprietária, que dados é que pretendem enviar e com que serviços querem partilhar os mesmos. Caso a confiança na *cloud* fosse assegurável, não seria possível partilhar resultados da agregação de dados dos dispositivos com os serviços apropriados sem que os mesmos dados fossem revelados, ou seja, os serviços teriam acesso integral a todos os dados (uma vez que, nesta arquitetura, a *cloud* fornece os dados aos serviços juntamente com a chave secreta), o que não é desejado.

Em [63], os autores propuseram uma ideia de uma arquitetura IoT que fornece segurança e avalia a qualidade dos dados que são recebidos de uma determinada fonte através de anotações e mecanismos semelhantes. No entanto, este foca-se muito mais na análise de qualidade do que no estabelecimento de métodos de segurança que reforcem a privacidade dos utilizadores. Os autores não mencionam mecanismos nenhuns para a garantia da privacidade dos dados. Apenas referem que os dados necessitam de ser cifrados e decifrados de alguma forma para que, posteriormente, se possam efetuar anotações aos mesmos e analisar a sua qualidade utilizando um modelo específico de qualidade dos dados. Após serem recolhidos os dados provenientes dos dispositivos (sensores) é analisada a origem dos dados e a sua reputação, juntamente com o tipo de dados. É então atribuída uma pontuação para os diversos aspetos de segurança (integridade, autenticidade, confidencialidade e privacidade) e os dados são posteriormente anotados para futura referência.

Foram estudados, também, avanços feitos na questão do anonimato dos dados provenientes de dispositivos. A privacidade dos dados pode ser reforçada, também, com a implementação de mecanismos de anonimato. Isto é, mecanismos que evitam que os consumidores dos dados obtenham informação suficiente sobre os produtores dos mesmos. Para tal existem mecanismos como o k-anonimato [64]. No entanto, este mecanismo foi desenvolvido para que seja possível efetuar o anonimato de dados apenas com um conjunto de dados estáticos. O k-anonimato permite que, na existência de dados específicos sobre um determinado campo ou tipo de informação (por exemplo a idade de uma pessoa), revelar outros dados onde existe a garantia que indivíduos cujos dados lhes são aplicados (por exemplo indivíduos que têm 25 anos) não podem ser identificados. Estes novos dados revelados são ditos que possuem a propriedade de k-anonimato caso, pelo menos, k indivíduos estejam incluídos na informação revelada. Um exemplo: se numa base de dados existirem exatamente 5 entradas de pessoas com atributo de 25 anos de idade e 5 pessoas com 24 anos de idade e assumindo que $k = 10$,

sempre que se necessite de obter a idade de uma destas pessoas, é seguro retornar um resultado "entre 24 e 25" anos. Assim, 10 pessoas podem identificar-se apenas divulgando este resultado. Os dados mais específicos das idades de cada pessoa na base de dados seriam então substituídos pelos novos dados gerados, mais abrangentes (generalizados).

A informação que os dispositivos a mais baixo nível produzem e enviam para a plataforma IoT são dados contínuos em tempo real. Estes dispositivos produzem, tipicamente, uma grande quantidade de dados num curto intervalo de tempo que são enviados para a plataforma IoT. Para conseguir um anonimato destes dados é necessário utilizar um outro mecanismo. Os autores de [11] propuseram o CASTLE (Continuously Anonymizing Data Streams). Este esquema consiste em *clusters* que agrupam tuplos. Estes tuplos contêm vários atributos (semelhantes a uma entrada numa base de dados relacional). Retomando os exemplos de pessoas, imaginemos que existem 3 pessoas com 24, 25 e 30 anos de idade respetivamente. O tuplo seria (nome, idade). Seria possível, com esta informação, formar um *cluster* de tamanho 3 C cujo intervalo de idades é $[24, 30]$. Assim que um *cluster* chegar a um valor k, isso implica que existem 3 pessoas distintas cujos tuplos pertecem a C.

Inicialmente, não existem *clusters* em memória. Quando um tuplo é recebido, um *cluster* é, então, criado. à medida que vários tuplos chegam, esse tuplo é incluído num determinado *cluster* caso o os valores do tuplo se possam incluir na respetiva gama de valores do *cluster*. Se não for o caso, o *cluster* é alargado o suficiente de forma a que o tuplo se possa inserir. Quando os tuplos crescem demasiado, os *clusters* são divididos. O CASTLE lida ainda com problemas de sobreposição de *clusters*, otimização no alargamento de *clusters*, garantia que o tamanho dos clusters é, pelo menos, k, substituição direta nos tuplos que chegam pela generalização do *cluster*, tem em conta atrasos (tempo desde que a informação chega para ser processada até estar pronta para ser libertada com anonimato), etc. O principal problema no uso deste tipo de mecanismos de anonimato de dados centra-se na grande perda de informação. Mesmo que a privacidade dos dados seja mantida e respeitada, há o risco do anonimato dos dados ser demasiado destrutivo, não sendo possível obter resultados precisos e fornecer dados com uma granularidade suficientemente elevada para que seja aceitável o uso dos mesmos pelos serviços/aplicações no contexto do IoT. Assim, não nos podemos restringir, exclusivamente, na aplicação deste método para poder conseguir um acesso a dados suficientemente granular e que preserve a privacidade dos produtores no que toca aos seus dados. Para além disto, os dados chegam em claro à entidade responsável por armazenar e aplicar estes mecanismos sobre os dados, pelo que esta entidade torna-se num risco à privacidade dos utilizadores, pois tem acesso integral ao conteúdo do dados.

Em [21] apresenta-se uma arquitetura no contexto da IoT onde são utilizados

dispositivos (sensores) médicos na obtenção de dados. O objetivo do trabalho é que seja possível proteger estes dados ao nível da confidencialidade desde que estes são gerados nos sensores médicos até que cheguem às aplicações médicas e serviços presentes, como é normal em arquiteturas de plataformas IoT, na borda de toda a rede. A cifra de dados nó-a-nó introduz vulnerabilidades no que toca à proteção contra possíveis invasores. Isto porque este tipo de cifra em qualquer que seja a arquitetura, implica que os dados, forçosamente, em alguma altura, terão de estar armazenados (nem que seja em RAM) em claro em um dos nós. No caso de uma arquitetura IoT, os dados teriam de se encontrar em claro nalgum dos componentes. Para além de ser uma solução que exigiria a confiança total nos próprios nós por onde passam os dados, seria uma catástrofe caso algum atacante se apoderasse, ou encontrasse alguma vulnerabilidade no acesso, de algum dos componentes da plataforma, pois teria acesso a todos provenientes dos sensores. Como sabemos, os dados médicos são de cariz privada e é uma violação da privacidade de um paciente caso pessoas não autorizadas tenham acesso aos mesmos. Assim, é necessária uma cifra de dados extremo-a-extremo.

A arquitetura apresentada consiste em três principais componentes: os sensores médicos móveis, as *gateways* típicas de uma plataforma IoT e a estrutura de retaguarda que tratará do armazenamento dos dados e dos serviços/aplicações que os pretendam utilizar. Para esta estrutura de de retaguarda foi utilizado um serviço de *cloud*. Os sensores enviam todos os dados para o *gateway* a que está ligado/associado. O *gateway*, na esmagadora maioria dos casos, é um dispositivo que possui maiores capacidades de armazenamento, memória e processamento relativamente aos dispositivos que lhe estão associados. Assim, é possível aplicar mecanismos de cifra dos dados apenas neste componente, ao invés de os aplicar nos próprios dispositivos geradores de dados, no entanto é necessário considerar que existe um perímetro de confiança. Neste caso, esse perímetro envolve os dispositivos e os *gateways*. Relativamente aos mecanismos e processos para efetuar a cifra, os autores não foram muito específicos e apenas mencionaram que seria necessário aplicar uma cifra utilizando a chave pública da aplicação da *cloud*, previamente presente no *gateway*, aos dados e enviá-los para a mesma através da Internet. Esta arquitetura não é, claramente, suficiente para que se consiga obter a privacidade desejada dos dados provenientes dos sensores. Uma vez que as aplicações de *cloud* tem acesso aos dados em claro (decifrando-os com a sua chave privada), a mesma teria que ser confiável, o que é algo a evitar, uma vez que a confiança não protege os dados de serem extraviados ou até mesmo utilizados sem o consentimento dos utilizadores/pacientes/produtores dos mesmos dados. Para além disso, os sensores teriam de estar vinculados a uma aplicação da *cloud*, uma vez que a sua chave pública necessitaria de estar pré-armazenada no sensor, não existindo quaisquer mecanismos de obtenção e armazenamento de chaves.

Para além da arquitetura, os autores analisaram também o desempenho do *gateway* em cifrar os dados. Obtiveram resultados de menos de 1 segundo na cifra utilizando uma chave pública. Concluindo, a principal preocupação dos autores deste trabalho foi na proteção dos dados através da Internet para que cheguem com segurança à aplicação final. No entanto, as aplicações/serviços não devem ter sequer acesso a todos os dados sensíveis e classificados privados de um utilizador (dono e produtor dos dados). Isto porque os dados são fornecidos ao consumidor exatamente como foram produzidos e sem nenhuma camada de abstração ou anonimato sobre os mesmos, mesmo considerando que se estão a proteger dados médicos onde toda a precisão é fundamental para o bem estar do paciente em questão. O consumidor, ao obter informação tão detalhada sobre os utilizadores está a pôr em causa a privacidade dos mesmos. Nesta dissertação não nos focaremos tanto nos dados médicos, no entanto, o trabalho desenvolvido será passível de ser utilizado também no contexto médico, dependendo das situações e dos requisitos que a medicina exija sobre a granularidade dos dados e atrasos ente a produção e chegada de dados, nomeadamente no caso em que são efetuados certos estudos sobre um histórico de dados de um utilizador.

Em [38], os autores apresentaram o modelo de um sistema baseado em *cloud* (Platform as a Service (PaaS)) cujo objetivo seria proteger e manter a privacidade de *software* provenientes de dispositivos clientes. Esta arquitetura não é orientada à IoT, no entanto são aplicados mecanismos de segurança para manter a privacidade dos dados enviados. A arquitetura consiste na existência de coprocessadores criptográficos que oferecem segurança a eventuais dados geridos pelo mesmo. Este *hardware* é resistente a ataques e interferências físicas, não permitindo que se efetuem operações não autorizadas, que se abra o dispositivo e que sejam concluídos com sucesso ataques por canal lateral. Para explicar este tipo de ataques, este será comparado com ataques de força bruta (*brute-force*) e com a análise teórica de algoritmos criptográficos (criptanálise).

Os ataques de força bruta consistem na constante, sistemática e exaustiva tentativa de uso de muitas (ou todas) as combinações de chaves num sistema criptográfico, ou senhas, por exemplo, num sistema de *login*, até que a chave correta seja encontrada. No caso de um ataque de força bruta o comprimento da chave é vital para que o atacante possa analisar se tal ataque seria viável. Isto acontece porque, quanto maior for o tamanho da chave utilizada, o número de combinações possíveis sobe exponencialmente, aumentando o tempo que o ataque demoraria, em média. No melhor caso, a primeira chave na lista de combinações a testar contra o sistema/algoritmo criptográfico seria a correta e não seria necessário prosseguir a análise do resto da lista de combinações. No pior caso, a última chave a ser testada seria a correta e envolveria o teste de todas as combinações possíveis de uma chave com determinado tamanho antes que fosse encontrada a chave correta. No caso de senhas, o mesmo acontece, no entanto, o

atacante pode utilizar apenas combinações de caracteres que formam senhas comuns ou presentes num dicionário ou lista (ataque com dicionário, *dictionary attack*).

A análise teórica de algoritmos criptográficos consiste na análise de dados em claro e na análise de dados cifrados com uma chave de forma a encontrar vulnerabilidades e fraquezas no respetivo sistema/algoritmo criptográfico. Um atacante é bem sucedido se conseguir descobrir padrões ou mecanismos que permitam decifrar dados sem o uso de chaves ou através de manipulação dos diferentes tipos de dados de alguma forma. Caso tal aconteça num determinado sistema/algoritmo significa que foi encontrada uma brecha/vulnerabilidade de segurança e todos os dados que utilizaram esse sistema ou algoritmo correm, agora, o risco de serem decifrados facilmente, anulando a característica de confidencialidade que tinha sido atribuída aos dados.

Os ataques por canal lateral, ao contrário dos ataques de força bruta e e ao contrário da análise teórica de métodos criptográficos, envolvem informação obtida da implementação física de um sistema criptográfico. Informação sobre o tempo que o sistema demora a processar a cifra/decifra dos dados pode ser do interesse do atacante, uma vez que pode revelar que tipo de chave (utilizando características do seu tamanho, por exemplo) está a ser utilizada. Informação sobre o consumo de energia pode também tornar-se útil para um atacante que, similarmente à informação do tempo, ajuda a revelar chaves e métodos usados na criptografia. Outras informações como análise do som, introdução de falhas propositadas, análise de dados que deveriam ter sido eliminados após mecanismos de cifra, etc. são informações puramente físicas que ajudam a revelar ao atacante o que está a acontecer dentro do sistema criptográfico.

Os coprocessadores usados neste trabalho são seguros contra este tipo de violações físicas e são instalados diretamente na infraestrutura *cloud* em cada servidor. Uma vez que um coprocessador por utilizador não é economicamente viável, os autores propuseram que cada servidor executasse uma máquina virtual e que os utilizadores partilhassem coprocessadores criptográficos. Para efetuar esta partilha é necessária a existência de um TTP para efetuar o armazenamento de pares de chaves (chaves públicas e privadas) para os diferentes utilizadores nos coprocessadores criptográficos. Em cada co-processador está a ser executado um *daemon* de raíz e que é o único que executa funções criptográficas sobre os dados, sendo todo este ambiente seguro. Existem ainda ambientes não seguros no coprocessador para efeitos de processamento de dados cuja sensibilidade não justifique a sua proteção. Esta diferenciação dos dados ao nível da privacidade é enviada para a *cloud* a partir dos dispositivos produtores que classificam os dados.

Uma desvantagem deste tipo de mecanismos é que apenas é possível fazer uso de uma plataforma na *cloud* para armazenar os dados de forma privada, não suportando mecanismos não seguros de armazenamento de dados produzidos. Desta forma, mesmo

os dados que não necessitam de quaisquer mecanismos de privacidade não podem ser considerados como tal, resultando num desempenho inferior para qualquer que seja o tipo de dados, no acesso aos mesmos. A TTP é vista como um vendedor de um serviço de *cloud* que oferece mecanismos de privacidade. Este facto leva a que a plataforma IoT tenha de ser específica. A *cloud* sozinha não consegue aceder aos dados por si só, no entanto, se cooperar com a TTP, tal pode acontecer, pois este tem acesso sobre a operação dos diferentes coprocessadores e tem posse dos pares de chaves gerados, podendo resultar em brechas no acesso aos dados.

No momento de registo dos utilizadores com o serviço de *cloud*, são gerados pares de chaves que são, como referido, injetados nos coprocessadores pela TTP. No entanto, uma cópia dos pares de chaves assimétricas são enviadas para o dispositivo produtor do utilizador através de uma ligação "segura". A partilha de chaves privadas não deve ser efetuada e este tipo de procedimentos causa um grande risco de acesso indevido aos dados por parte de terceiros, caso estes se apoderem das chaves a serem transmitidas (*eavesdropping*). A chave privada pertencente ao utilizador (dono dos dados) nunca poderia ser transmitida desta forma nem deveria, sequer, ter sido gerada pela TTP, uma vez que facilita o seu acesso indevido aos dados. Os autores mencionam que uma possibilidade seria efetuar esta transferência de chaves a nível pessoal. No entanto, tal não seria prático nem suportável com o aumento de utilizadores. Os autores referem também que a TTP pode atualizar remotamente, por questões de gestão, os pares de chaves presentes nos coprocessadores. Esta atualização remota é um outro problema. Uma vez que a frequência de transmissão de chaves é alta, os atacantes poderão estar mais atentos às ligações da TTP aumentando a probabilidade de acesso às chaves.

Os produtores dos dados (dispositivos/sensores) cifram os dados utilizando uma chave simétrica gerada no momento. Essa chave simétrica é cifrada com a chave pública do *daemon* de raíz a correr no coprocessador (cifra híbrida). Assim, só o coprocessador pode aceder aos dados. Juntamente com esta informação, são enviadas assinaturas de toda a mensagem incluindo alguns meta-dados. Uma vez que se trata de uma PaaS, o artigo foca-se mais na transmissão de *software* privado para ser executado nos coprocessadores. No entanto, este *software* pode ser considerado como informação (dados comuns) proveniente de sensores. O *daemon* de raíz dos coprocessadores procede então à validação das assinaturas e à decifra e armazenamento/execução dos dados/*software*. O resultado da execução do *software* no dispositivo necessita agora de ser enviado de volta ao requerente. Isto equivale, na plataforma IoT em obter os dados por parte do consumidor. No entanto, neste caso, os dados são obtidos pelo produtor dos mesmos. Uma vez que tal é o caso, os restantes passos e fluxo de dados torna-se irrelevante para que sejam aplicados numa plataforma IoT, uma vez que não existe partilha de dados com entidades que lhes pretendam aceder.

Em [68] as plataformas IoT foram analisadas com o mesmo intuito do trabalho anteriormente mencionado. Ou seja, o objetivo seria obter informação dos mecanismos que estão, neste momento, a ser aplicados para garantir a segurança dos dados no momento do armazenamento numa *cloud*. No entanto, este trabalho apenas apresenta uma solução de aumento da segurança dos dados no armazenamento utilizando auditoria dos mesmos. A inovação aqui é que a entidade que valida/audita os dados, não precisa de lhes aceder nem de manter uma cópia local. Para tal é envolvida uma TTP. Os mecanismos presentes neste trabalho podem ser interessantes para aumentar a segurança sobre os dados armazenados mantendo também a privacidade dos donos dos mesmos. Contudo, este é um problema adicional ao que se está a resolver nesta dissertação e, por isso, não entraremos em detalhes. Para além disso, este trabalho apenas se foca na produção e acesso aos dados por um mesmo interveniente. Nesta dissertação procura-se encontrar soluções de partilha de dados não pondo em causa a privacidade do utilizador.

Em [39] procura-se resolver problemas de segurança e privacidade em *smart grids* que explore *smart meters*. Um *smart meter* (ou dispositivo de medida inteligente) é, como referido na Secção 3.1, um dispositivo que contém um contador/medidor de eletricidade, água ou gás que é ligado a um microcontrolador. Os dados vão sendo contados e enviados, através de um *gateway*, para a Internet. O dispositivo de medida e o *gateway* podem ser apenas um componente presente apenas numa caixa, ou não. Caso não seja, a sua comunicação tem de ser protegida ou assumida como confiável. Uma *smart grid* consiste numa plataforma onde chegam dados provenientes dos *smart meters* de várias propriedades. Estes dados são depois acedidos por aplicações também ligadas à rede. Esta arquitetura é em tudo semelhante a uma arquitetura IoT, pois possui os seus três principais componentes: consumidores, uma plataforma onde são guardados os dados e aplicações/serviços que lhes acedem.

O problema da privacidade dos dados de contadores/medidores elétricos foi especificado e explicado na Secção 3.1 em dois trabalhos diferentes.

Com o objetivo de manter a privacidade dos dados, os autores apresentaram 4 soluções. A primeira solução, classificada de solução óbvia, é aumentar o intervalo de envio de dados para fora. Esta solução resolveria o principal problema dos dados medidos: análise de padrões de consumo com o objetivo de obter informações privadas sobre o utilizador alvo. O problema desta solução é que se o intervalo dos dados for demasiado grande, muita informação, que poderia ser relevante para as aplicações/serviço que consomem os dados, perder-se-ia e poderia não corresponder aos requisitos necessários dessas aplicações/serviços para que funcionassem em condições ou com alguma utilidade.

A segunda solução, classificada de solução estatística, consiste na inclusão de um TTP ou um agregador de dados que efetua o anonimato dos dados e fornece uma vista sobre os mesmos às aplicações/serviços que lhes pretendam aceder. Assim, as aplicações

poderiam obter detalhes mais precisos, no entanto, esses dados não pertenceriam apenas a um cliente/propriedade, pois os dados seriam generalizados. O problema desta solução está na existência de um TTP. Se este for comprometido, os dados estarão em risco. Os autores não falam de métodos de cifra de informação nas comunicações com esta suposta TTP que, a existir, teriam de ser implementadas para garantir a segurança dos mesmos. Para que a privacidade dos dados pudesse ser mantida, quer o cliente quer a empresa necessitariam de confiar totalmente na TTP. Um outro problema reside no requisito da existência de clientes/propriedades suficientes e com dados do contador semelhantes para que a estratégia de anonimato pudesse ser aplicada (k-anonimato).

A terceira solução, classificada de solução matemática, é a mais elaborada e consiste na adição de distorções aos dados medidos aleatoriamente antes que estes sejam enviados para fora do *gateway*. Estas distorções seriam aplicadas de tal forma que, no momento da agregação/combinação de todos os dados individuais recolhidos, as diferentes distorções aplicadas anular-se-iam. Imaginemos que existem n valores aleatórios p_i cujo somatório é zero ($\sum_{i=0}^{n} p_i = 0$). Cada medidor transmitiria, assim, um valor distorcido c_i processando o valor medido m_i e aplicando uma distorção p_i: $c_i = m_i + p_i$. No momento de recolha dos dados na aplicação, esta pode combinar todos os valores distorcidos recebidos e obter uma soma final destes que corresponde a uma soma final dos dados originais: $\sum_{i=0}^{n} c_i = \sum_{i=0}^{n} m_i + p_i = \sum_{i=0}^{n} m_i$. Ao invés de ser utilizado um somatório simples par efetuar as distorções, poderia ser usado outro método qualquer de distorção de dados, como por exemplo utilizando os expoentes de Diffie-Hellman, como referem os autores. Existe uma grande vantagem neste método que é o facto de não ser necessário qualquer TTP, pois os dados enviados pelo próprio medidor podem ser utilizados. No entanto, para além de ser um mecanismo um pouco complexo, ainda peca no facto de ser necessária a receção e análise das n medidas para que seja possível obter valores fiáveis. Outro problema no uso deste mecanismo é que apenas podem ser tratados dados numéricos provenientes dos sensores, enquanto no contexto do IoT, não existem apenas medidores e contadores de água, energia e gás. Também existem outros tipos de dados cuja proteção de privacidade não pode ser resolvida através desta solução.

Por fim, a última solução apresentada pelos autores é manter os dados medidos privados em disco ou *buffer* de dados localizados junto dos donos dos dados. Quando fosse necessário obter os dados, alguém teria de efetuar a recolha destes equipamentos, o que resultaria numa maior despesa económica por parte dos fornecedores de energia. Para além disso, esta solução não faria qualquer uso de uma plataforma IoT.

Os autores identificaram ainda diversos tipos de problemas de segurança em *smart grids* e mencionaram possíveis soluções: a existência de *eavesdroppers* que tomam posse dos dados acedendo ao meio por onde eles passam, ataques *man-in-the-middle* com o

objetivo de alterar ou danificar os dados que passam desde os medidores até às aplicações e injeções de comandos remotos falsos - estes problemas podem ser resolvidos através da implementação de mecanismos de cifra nas comunicações; ataques na infraestrutura de armazenamento de dados ou comprometimento da integridade do medidor inteligente pode ser resolvido através da implementação de mecanismos de verificação de integridade do sistema, *sandboxing* de máquinas virtuais ou sistemas de deteção de intrusos. Por fim, um problema de difícil resolução que assombra não só as *smart grids* e a IoT mas também toda a infraestrutura da Internet atual: ataques Distributed Denial of Service (DDoS).

Os autores de [60] tiveram como alvo, assim como no trabalho anterior, as *smart grids*. Neste trabalho são apresentados mecanismos que prometem a análise da "quantidade" de privacidade dos dados medidos nos contadores/medidores ao ser aplicada uma distorção. Os autores apresentam de forma muito detalhada o processamento que fazem sobre os dados recorrendo a inúmeras fórmulas, cálculos, teoremas e definições matemáticas, pelo que não entraremos em detalhes sobre o seu processo.

Neste trabalho é proposta uma solução semelhante à ideia da distorção dos dados ao nível dos medidores antes de ser feito o envio dos mesmos para fora da rede, que foi apresentado no trabalho anterior. Os dados são perturbados (termo utilizado mais frequentemente no trabalho) de forma a garantir um determinado nível de privacidade. Ao mesmo tempo, é necessário obter um determinado nível de fidelidade/utilidade. Este trabalho propõe um sistema que calcula o equilíbrio entre utilidade e privacidade, pois quanto maior é a utilidade (menor perturbação criada nos dados), menor é a privacidade e quanto maior é a privacidade (maior perturbação criada nos dados), menor é a utilidade. O modelo desenvolvido pelos autores permite calcular o nível de informação privada liberada e o nível de utilidade dos dados independentemente do tipo de distorção aplicada. Uma aplicação desta *framework* conjuntamente com um método de distorção é muito útil caso a distorção dos dados pelos medidores seja utilizada como mecanismo de garantia de um certo nível de privacidade e utilidade. Assim, seria possível garantir que os dados enviados para a plataforma subjacente, seriam, de alguma forma, suficientemente privados, não sendo necessário aplicar métodos mais complexos sobre os dados nem sendo necessária a inclusão de uma TTP na arquitetura.

O objetivo de [24] foi o de resolver o problema da privacidade dos dados de dispositivos que produzem os dados e os armazenam na *cloud*. Os autores oferecem a possibilidade de transparência no envio de dados para a *cloud*, uma vez que múltiplos serviços *cloud* podem ser utilizados e não existem dependências de fornecedores deste tipo de serviços. A arquitetura proposta está ilustrada na Figura 3.3.

Como podemos verificar, a arquitetura possui uma separação sobre o conjunto de componentes que necessita de ser confiável e o conjunto de componentes onde não existe

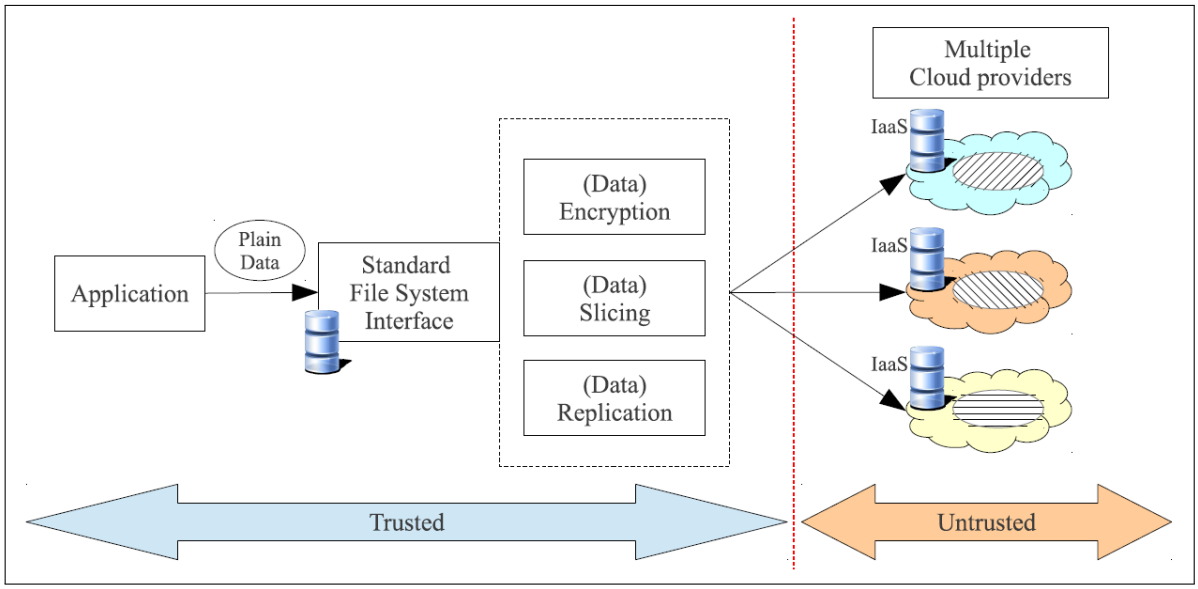

Figura 3.3: Arquitetura de sistema transparente para armazenamento na cloud *proposta por [24]*

confiabilidade e, portanto, os dados necessitam de ser protegidos ao navegarem para esta zona. Os autores assumiram que os dados seriam armazenados na *cloud* utilizando o paradigma Infrastucture as a Service (IaaS), no entanto, o modo de armazenamento dos dados é irrelevante e pode depender de implementação para implementação.

Uma determinada aplicação (que no contexto do IoT pode ser equiparada a um dispositivo produtor de dados) necessita de armazenar os dados que produz remotamente na *cloud*. Os seus dados são enviados para um sistema de ficheiros local ao dono dos dados. Este sistema de ficheiros gere todos os dados que recebe de forma segura. No contexto do IoT, poderíamos equiparar este componente a um *gateway* com a única diferença que, no *gateway*, não seriam armazenados (ou seriam armazenados temporariamente) dados. O *gateway*, após efetuar a receção dos dados, aplicaria de imediato os mecanismos de garantia de segurança propostos pelos autores e enviaria informação diretamente para a plataforma IoT (ou *cloud* no contexto do trabalho em questão), sem que os dados cifrados e outros meta-dados fosse armazenados. Os autores propõem que o sistema de ficheiros possua uma interface padrão para que sejam recebidos dados. Após a receção, os dados seguem para um módulo de cifra que cifra os dados utilizando *Dm-Crypt* (um sistema de cifra de dados em disco, que faz uso do AES utilizado por versões recentes do sistema operativo Linux). Os dados cifrados são depois armazenados num sistema de ficheiros distribuído. Este componente trata de dividir os dados cifrados por diferentes fornecedores de serviços *cloud* e efetua ainda replicação. Para comunicar com a *cloud*, a arquitetura ainda permite a adição de uma VPN, por forma a aumentar o grau de confidencialidade no transporte dos dados.

Esta arquitetura é uma boa arquitetura para garantia de privacidade de dados na *cloud*, pois a utilização de diferentes serviços de *cloud* para efetuar o armazenamento de dados cifrados e repartidos é uma excelente forma de aumentar o grau de privacidade proporcionado ao utilizador, pois seria mais difícil efetuar um ataque sobre os mesmos: para que fosse possível obter um determinado conjunto de dados, seria necessário o atacante obter porções de dados a partir das diversas infraestruturas *cloud* e ainda decifrar os dados onde é necessária uma chave que nunca saiu para fora do dispositivo que a gerou. No entanto a mesma serviria apenas para uso pessoal, uma vez que não existe partilha de dados entre diversas entidades. Assim, tal mecanismo não poderia ser aplicado ao IoT onde existem diferentes entidades que produzem e consomem os dados.

Em [36] os autores propuseram uma arquitetura focada no utilizador (aquele que tem em sua posse os dispositivos produtores de dados) de forma a que este possa, intuitivamente e através de ficheiros de configuração, controlar que dados são enviados e quais as condições impostas no seu envio. Para além do controlo sobre os dados recolhidos pelo dispositivos (sensores), os autores desenharam uma arquitetura que protege a privacidade dos dados do utilizador no controlo de acesso. É assumido que toda a plataforma IoT pode ser curiosa no que toca ao acesso aos dados, no entanto o seu funcionamento não é modificado, na realização deste trabalho.

A arquitetura de segurança apresentada não envolve terceiros. Os componentes existentes numa plataforma IoT padrão são alterados sendo que estes passam a efetuar processamento de dados diferentes e comunicações diferentes com os componentes vizinhos. Apenas é necessário definir a existência de um produtor que envia dados para uma infraestrutura. Diferentes aplicações/serviços/entidades poderão depois aceder a esses dados construindo *queries*. Assumiremos que o produtor de dados mencionado no trabalho é constituído por um dispositivo medidor/produtor e por um *gateway* (dispositivo com maiores capacidades de processamento e armazenamento relativamente aos produtores, e que encaminham os dados recebidos).

Inicialmente o produtor dos dados envia para a infraestrutura IoT os dados cifrados com uma chave simétrica gerada e outra informação como identificadores e *hashes* de conteúdo para efeitos de validação de integridade. O método de cifra aqui utilizado é apenas um conjunto de XORs entre a chaves simétrica e os porções dos dados. A cifra utilizando apenas o XOR não é recomendada uma vez que esta possui fraquezas. Primeiramente, caso a chave seja mais pequena do que os dados, é possível determinar o seu comprimento através de métodos de criptanálise: aplicar o XOR entre os dados cifrados e os mesmos dados cifrados com um determinado deslocamento. Experimentando vários valores de deslocamento e analisando a semelhança entre os *bytes* resultantes e o texto cifrado original, é possível obter informações sobre o tamanho da chave. Caso a percentagem de *bytes* iguais seja acima de um certo limiar, é extremamente provável que

o número de deslocamentos executados sobre os dados seja um múltiplo do tamanho da chave. Ao encontrar o número mínimo de deslocamentos que resulte num grande número de *bytes* iguais, estamos também a encontrar o tamanho da chave. Agora que já é conhecido o tamanho da chave, é possível efetuar um XOR entre o texto cifrado original e o texto cifrado original deslocado tantas vezes quanto o tamanho da chave. A chave é anulada e o resultado é o XOR entre o texto em claro e o texto em claro deslocado o que já permite obter alguns segmentos de dados muito semelhantes que, em muitos casos, podem revelar o conteúdo dos dados.

O fluxo dos dados envolvem muitos passos e muita troca de informação, identificadores e *hashes* pelo que não serão aqui detalhados os pormenores acerca de toda a comunicação existente entre os diferentes componentes. De um modo geral, existe uma troca de informação acerca de identificadores e *hashes* entre o consumidor e o produtor. Na mesma mensagem que o consumidor troca com o produtor (através da infraestrutura IoT que procede ao reencaminhamento das mensagens), é enviado um pedido à infraestrutura de acesso aos dados. A infraestrutura responde com os dados cifrados e com *hashes* para o mesmo efeito descrito anteriormente. Na fase de revelação dos dados cifrados ao consumidor, primeiramente são trocadas mensagens entre o produtor e o consumidor através da infraestrutura com troca de strings aleatórias *hashes* e sua validação para efeitos de autenticidade e integridade das mensagens trocadas. Uma comunicação adicional entre o produtor e o consumidor é efetuada através de canais protegidos para que seja concluída com sucesso a validação das mensagens trocadas. Para que seja possível decifrar os dados, são ainda trocadas chaves simétricas utilizando uma abordagem assimétrica (utilização da cifra assimétrica juntamente com cifra simétrica - cifra híbrida).

Nesta arquitetura, quando se fala de produtores, na verdade está-se a referir aos *gateways*, uma vez que os dispositivos produtores nunca poderiam estar encarregados de fazer processamento criptográfico sobre os dados, devido à sua baixa capacidade, comum neste tipo de *hardware*, de armazenamento e processamento. Mesmo admitindo tal, existe ainda um outro problema relativamente à segurança dos dados. Os consumidores, ao decifrarem os dados, iriam ter acesso aos mesmos. Logo, ou todos os consumidores teriam que ser confiáveis como um requisito (o que não faz qualquer sentido num sistema deste género), ou então estes poderiam utilizar os dados produzidos na camada mais baixa da rede para outros fins que não os que foram acordados com o utilizador. Se tal acontecesse, não existiria qualquer privacidade nos dados, uma vez que estes já se encontram à disposição de outros, sujeitos a cópias, transferências, vendas, etc. dessa informação. Para combater este facto, os autores dão a possibilidade aos utilizadores da sua plataforma de, através de ficheiros de configuração, estabelecerem regras e condições para a transferência de dados. Neste tipo de abordagem está incluída a configuração do

uso de k-anonimato conjuntamente com papéis cientes de contexto.

O k-anonimato foi explicado anteriormente. Papel ciente de contexto (Context Aware Role) é uma função baseada no contexto de comunicação entre entidades com um determinado *role*. O papel (*role*) é também uma função baseada em semânticas relacionadas com a autoridade e responsabilidade conferidas no consumidor associado a um papel. A combinação entre estes dois mecanismos pode ser utilizada, por exemplo, para estabelecer um certo grau de anonimato dependendo das condições de contexto em que determinada entidade se encontre, fornecendo um grande controlo ao utilizador sobre os dados que são transmitidos pela rede. No entanto, como referido anteriormente nesta dissertação, dependendo do grau de anonimato conferido a certos dados considerados privados, pode ser possível que: caso o grau de anonimato seja grande, os dados não sejam muito úteis para o consumidor; caso o grau de anonimato seja baixo, os dados do produtor não sejam privados. Logo, seria necessária a existência de um acordo entre o produtor e o consumidor que definisse o equilíbrio entre o grau de utilidade e o grau de privacidade dos dados a serem transmitidos pela rede. Caso esse acordo não se consiga estabelecer, os dados nunca estariam privados e seguros nesta arquitetura.

3.2.2 NOVOS MÉTODOS CRIPTOGRÁFICOS DE SEGURANÇA

Assim como existe a evolução de tecnologia, também existe evolução e progressos feitos na área da segurança, nomeadamente na criação de novos mecanismos de segurança para além dos tradicionais, como os mecanismos de cifras simétricas, cifras assimétricas, funções de *hashing*, códigos de autenticação de mensagens, entre outros. Nesta secção serão analisados os métodos utilizados e propostos por diferentes trabalhos.

Os autores de [59] afirmam resolver o problema da privacidade dos dados quando enviados para uma infraestrutura/plataforma *cloud* de forma a que estes possam ser partilhados com outros. No contexto do IoT, os donos dos dados que pretendem guardar os seus dados remotamente são equiparados aos dispositivos produtores de dados, a *cloud* é equiparada a uma plataforma IoT e os utilizadores (produtores de dados) com os quais foi efetuada a partilha dos dados são equiparados aos consumidores dos dados.

Neste trabalho foi desenvolvida uma plataforma cujo objetivo é aplicar mecanismos de segurança sobre os dados de forma a que estes possam ser partilhados com terceiros que possuam as permissões necessárias. Para além disso, a plataforma também possui algumas funcionalidades como revogação e *re-join* de utilizadores para acesso aos dados e previne a existência de ataques em conluio entre diferentes entidades que desejem aliar-se com a única intenção de quebrar a segurança do sistema e dos dados. Os novos/avançados mecanismos de segurança utilizados pelos autores são a cifra

homomórfica e re-cifra baseado em *proxy* (*proxy re-encryption*).

A cifra homomórfica é um tipo de cifra que permite que sejam efetuadas, com segurança, um conjunto limitado de operações sobre os dados cifrados [58]. A segurança é garantida, pois os dados protegidos não necessitam de ser decifrados para que possa ser aplicada uma função sobre eles. O resultado dessa função é fidedigno e é um texto cifrado que pode ser mais tarde decifrado. Para a cifra dos dados são utilizados pares de chaves (pública e privada). Como exemplo de utilização, imagine-se que existem dois valores inteiros x e y que são necessários cifrar independentemente (podem, por exemplo, ser dados medidos a diferentes alturas por um contador de energia elétrica). Os dados são, então, cifrados independentemente usando uma determinada chave pública e são enviados para a *cloud* (C_x e C_y). A *cloud* não lhes pode ter acesso uma vez que não possui a chave privada pertencente ao par de chaves cuja chave pública foi usada para cifrar os dois valores, portanto estes dados estão seguros e não podem ser "vistos" por ninguém (a não ser que exista alguém que conheça a chave privada respetiva, no domínio da *cloud*). Com os dados em segurança na *cloud*, imaginemos que existe um consumidor não confiável que deseje, não de os visualizar por completo (o que iria comprometer a privacidade dos mesmos), mas assim obter uma vista ou uma transformação sobre os dados: transformação essa (confiável) que já poderia ser obtida sem comprometer a privacidade dos dados originais. Uma forma de efetuar essa transformação utilizando uma cifra RSA tradicional seria dar acesso à *cloud* para efetuar a decifra dos dados e então efetuar essa transformação reencaminhando o resultado para o consumidor. No entanto, a *cloud* teria acesso aos dados originais, o que resultaria na perda de privacidade dos dados. Com as cifras homomórficas, é possível efetuar essa transformação sobre C_x e C_y sem que estes necessitem de ser decifrados. Caso a transformação fosse uma simples soma, o resultado (cifrado) a ser devolvido para o consumidor dos dados seria $C_R = C_x + C_y$. O consumidor (imaginando que possui a chave privada necessária para efetuar a decifra) pode então decifrar C_R resultando em $R = x + y$.

No entanto, as transformações que podem ser aplicadas aos dados estão limitadas (na forma simples da cifra homomórfica) a somas e multiplicações por números inteiros sobre os dados cifrados, pelo que o uso deste mecanismo de cifra em ambientes como a IoT poderia não ser o indicado, devido aos diferentes tipos de transformações sobre os dados que poderia ser requisitados e também ao facto de que, num ambiente IoT genérico, os tipos de dados podem não ser apenas números, caso em que não se poderia aplicar este tipo de cifra.

Relativamente à recifra de dados utilizando um *proxy*, este mecanismo permite que um *proxy* semi-confiável T possa converter um texto cifrado sob a chave pública do utilizador A num texto cifrado sob a chave pública do utilizador B. Para que esta conversão possa ser efetuada, é necessário que o *proxy* tenha em sua posse uma chave

de recifra $rk_{pk_a \to pk_b}$. Esta chave de recifra é exclusiva ao par origem (chave pública de A) - destino (chave pública de B). pk_a e pk_b são as chaves públicas de A e de B, respetivamente. Os dados nunca são decifrados e o texto original nunca é revelado a T. Este apenas necessita de ter armazenada internamente informação acerca das chaves de recifra mapeadas ao respetivo par origem-destino de chaves. Sendo x dados que necessitam de ser cifrados, PRE a função de re-cifra que tem como argumentos os dados cifrados e a chave de recifra exclusiva do par origem-destino de chaves públicas de ambos os utilizadores e $E_{pk_a}(x)$ os dados x cifrados sob a chave pública de A pk_a: $PRE(E_{pk_a}(x), rk_{pk_a \to pk_b}) \to E_{pk_b}(x)$.

A arquitetura deste trabalho consiste em 3 componentes principais, semelhantes a outros trabalhos: um produtor de dados (utilizador) que armazena os seus dados na *cloud* e, neste caso, pode efetuar operações de revogação e *re-join* do acesso aos seus dados por parte de outros utilizadores específicos, uma infraestrutura/plataforma de armazenamento dos dados (a *cloud* em si) e os consumidores de dados (outros utilizadores) que acedem aos dados da *cloud* caso tenham permissão. O dono dos dados necessita de gerar chaves e efetuar a sua troca com os consumidores, sendo que este é um passo de inicialização. O dono dos dados gera dois tipos de pares de chaves: par de chaves do dono dos dados e pares de chaves para cada consumidor dos seus dados. Estas chaves de permissão de acesso necessitam de ser comunicadas aos consumidores. Esta comunicação de chaves privadas pode ser problemática, pois caso algum atacante se apodere das mesmas, tem o mesmo controlo sobre os dados "privados" tal como o consumidor que foi autorizado a aceder aos dados pelo dono dos mesmos.

Devido à grande complexidade existente na preparação anterior ao envio dos dados para a *cloud*, não entraremos em detalhes acerca do procedimento. De forma geral, o conjunto de dados a serem enviados para a *cloud* são divididos por certos atributos. Tomando o exemplo de uma base de dados de pessoas, os atributos seriam, por exemplo, o nome, a idade e a profissão da pessoa. Usando números gerados aleatoriamente, estes são adicionados aos valores dos atributos resultando em dados cujos atributos estão modificados. Os diferentes atributos são individualmente cifrados com a chave pública do dono dos dados, o utilizador A. Agora, A pode optar por partilhar um conjunto de atributos desses dados com o utilizador B (como, exemplo a idade e a profissão) ou não permitir o acesso de B a esses dados. É assumido que B tem já em sua possa o par de chaves gerado por A. A cria um *token* de acesso que B tem a determinados dados. No caso em que B tem autorização para aceder à idade e profissão de determinada pessoa, o *token* irá ser um tuplo com a identificação de B, uma chave de recifra com origem a chave pública de A e destino a chave pública de B e uma lista contendo valores cuidadosamente escolhidos para os atributos que B pode aceder. Estes valores são cifrados com a chave pública de B. Caso B não possa aceder a esses dados, o *token*

é nulo. Para cada utilizador que pode aceder a determinados dados (composto por um conjunto de atributos), A necessita de criar um *token*. No contexto do IoT onde a maioria dos dados é enviado em tempo real (intervalo de tempo curto entre envios), estar a efetuar toda esta criação de *tokens*, processamento de atributos e cifras de dados para cada valor medido, por exemplo, por um contador de eletricidade resultaria num enorme impacto no sistema em termos de desempenho e de sobrecarga da rede, pelo que esta arquitetura não pode ser utilizada neste contexto. Além do mais, esta arquitetura não é escalável, uma vez que quanto mais são as aplicações/serviços que pretendem aceder aos dados, maior (aumenta exponencialmente) é a quantidade de processamento a ser efetuado na zona produtora de dados, pois seria necessário efetuar todos estes procedimentos para cada valor medido para cada aplicação que lhe pretenda aceder.

Para efetuar o acesso aos dados, a *cloud* verifica a existência do *token* apropriado. Caso exista, é utilizada a recifra dos dados armazenados na *cloud* para que B possa decifrar. Uma vez que os atributos que estão na *cloud*, para além de estarem cifrados, estão também modificados, é utilizada a cifra homomórfica para efetuar uma soma entre os dados cifrados (após aplicação de PRE) e os valores cuidadosamente escolhidos e cifrados presentes no *token*. O resultado desta operação é enviado para B que pode decifrar os atributos a que pode aceder, pelo que os valores dos restantes atributos são irrelevantes, aleatórios e considerados "lixo". Para além do enorme decréscimo de desempenho se este mecanismo fosse aplicado em IoT, o dono dos dados teria que estabelecer uma relação de confiança com cada aplicação/serviço que pretendesse aceder aos seus dados. Caso não fosse estabelecida, como essas aplicações/serviços têm acesso aos dados na sua íntegra, tal como foram originados, a privacidade dos mesmos é anulada, pois estes continuam a estar na mão de terceiros que podem fazer o que bem entenderem com os mesmos, sem o consentimento do seu produtor.

Em [25], é apresentado um método chamado de cifra homomórfica completa. Como referido anteriormente, a cifra homomórfica permite a execução de operações em dados cifrados sem que seja necessário decifrá-los, logo não é necessário possuir uma chave. Foi, também, referido que as funções que se poderiam aplicar aos dados eram limitadas a somas e multiplicações por números inteiros. Contudo, o método de cifra homomórfica completa apresentado neste trabalho permite que uma função arbitrária possa ser aplicada sobre os dados. Devido à grande complexidade por detrás do mecanismo e também porque nos afastaríamos do tema da dissertação, não será descrito. Este tipo de cifra traria, então, vantagens pois permite que sejam executadas quaisquer funções sobre os dados cifrados, no entanto, o mecanismo não pode ser aplicado na prática devido ao excessivamente longo período de tempo que é necessário para que todas as operações criptográficas possam ser concluídas com sucesso. Para além do problema do desempenho da cifra homomórfica completa, existe ainda um outro problema no

esquema apresentado pelos autores. O problema adicional reside no facto de que as operações de cifra e avaliação (operação de aplicação das desejadas funções sobre os dados cifrados) introduzem ruído. Apenas a operação de decifra remove ruído, pelo que se forem aplicadas demasiadas funções sobre os dados cifrados, os mesmos podem-se tornar inutilizáveis aquando da decifra ou com a sua integridade afetada. Por isso, e uma vez que estes mecanismos ainda estão a ser estudados, a sua aplicação em arquiteturas reais não deve ser feita.

Existem inúmeras formas de efetuar a recifra de dados, como referido em trabalhos anteriores. Para além da recifra básica através de um *proxy* (Proxy Re-Encryption scheme (PRE)), existem implementações mais específicas baseadas nesta, incluindo: PRE baseada em tipos, PRE com chave privada, PRE baseada em identidades, PRE condicional, PRE baseada em atributos, PRE baseado em tempo e PRE usando um limiar. Todos estes métodos e as suas vantagens e desvantagens são apresentados em [37], embora com pouco detalhe.

Em [18], os autores introduzem um novo mecanismo de controlo de acesso flexível e comparam-no com o tradicional modelo de controlo de acesso. Este último envolve uma entidade (o utilizador) que precisa de aceder a dados. Este utilizador comunica com uma outra entidade (monitor de acesso) que irá permitir esse acesso ou não. Para efetuar essa decisão, o monitor terá de aceder a regras, condições, políticas ou papéis facilmente disponíveis e geralmente presentes no mesmo dispositivo. Caso o monitor declare que o utilizador que pretende aceder aos dados tem permissões para tal, este acede aos dados e reencaminha-os para o utilizador. O monitor pode implementar várias tecnologias de controlo de acesso, nomeadamente: RBAC - mecanismo que atribui um ou mais papéis a utilizadores onde os papéis têm uma série de proposições que limitam ou não o acesso acesso a determinados dados pelo utilizador inerente; Mandatory Access Control (MAC) - existência de vários níveis de permissões de acesso onde um deles é atribuído a cada utilizador. Este pode aceder a dados cujo nível não é mais alto que o seu; e Discretionary Access Control (DAC) - o sistema tem de conhecer cada utilizador individualmente e atribuir-lhe um conjunto de permissões para cada tipo de dados presentes num sistema.

Os autores destacam as desvantagens das soluções de cifra utilizando chaves assimétricas - grande sobrecarga na gestão de certificados e falta de escalabilidade deste mecanismo; e chaves simétricas - problema de distribuição *online* das chaves para efeitos de partilha de dados. Sendo assim, os autores desenvolveram uma solução que utiliza Attribute-Based Encryption (ABE) onde tais desvantagens não se verificam e onde, adicionalmente, não é necessário qualquer tipo de entidade terceira (TTP) para mediar o controlo de acesso.

O mecanismo consiste em cifrar dados direcionados a todos os utilizadores que

possuam determinados atributos. Como é facilmente analisável, isto permite cifrar os dados uma vez e oferecer acesso aos dados a múltiplos utilizadores (cifra um-para-muitos). Como só quem possua um determinado tipo de atributos pode aceder aos dados, pode-se afirmar que o controlo de acesso está embutido na cifra. Existem vários tipos de ABE, nomeadamente Key-Policy Attribute Based Encryption (KP-ABE) [26] e Ciphertext-Policy Attribute Based Encryption (CP-ABE) [9] cujo desempenho foi comparado em [69].

No KP-ABE um conjunto de atributos dos dados são embutidos nos dados cifrados e a política de acesso aos dados está embutida na chave privada dos utilizadores que pretendem aceder aos dados. Estes apenas podem decifrar os dados caso a política de acesso presente na sua chave privada seja igual à presente nos dados cifrados. Como exemplo de aplicação, imaginemos que é necessário cifrar dados de um conjunto de pessoas e dar acesso a um terceiro a alguns dos seus dados. Uma das pessoas contém os atributos idade e profissão com os valores 25 e Professor, respetivamente. O dono da base de dados onde se encontram essas pessoas, pode limitar o acesso a elas cifrando esse conjunto de pessoas com os atributos específicos: 25 no campo idade *e* (*AND*) Professor na profissão (política de acesso). Assim, apenas pessoas com aqueles atributos especificados (retirados da política de acesso) podem ser decifrados pelo terceiro.

No CP-ABE a política de acesso aos dados são embutidos no texto cifrado e os atributos de um terceiro que pretenda aceder aos dados estão associados à sua chave privada. Verifica-se, portanto, a troca de responsabilidades relativamente ao KP-ABE. O terceiro que precise de aceder aos dados pode decifrar o texto cifrado apenas se os atributos associados à sua chave privada satisfazem a política de acesso embutida nos dados cifrados. Tomando o mesmo exemplo da base de dados de pessoas anterior, o dono da mesma pode limitar o acesso às pessoas cifrando o conjunto de pessoas com uma chave derivada de uma política de acesso cujo texto cifrado possa ser decifrado apenas se a política de acesso se verificar (por exemplo, caso a profissão da pessoa seja Professor *ou* (*OR*) Investigador). Assim, apenas terceiros cuja chave privada está associada com aqueles atributos especificados (retirados da política de acesso) podem decifrar os dados.

Através do uso destes mecanismos de cifra é possível resolver variados problemas de segurança existentes atualmente, no entanto o seu método ainda está em desenvolvimento e existe muito trabalho que necessita de ser feito. O modelo não foi submetido a testes de segurança em grande escala e, por isso, o seu uso em ambientes reais não é recomendado. Para além deste facto existe ainda um outro problema, que é o tempo despendido para cifra e decifra de dados, que é consideravelmente grande quando comparado com os mecanismos de cifra atuais. Ainda assim em [18] optou-se por fazer uso do CP-ABE para implementar um controlo de acesso flexivel em dados armazenados na *cloud*.

Em [34] os autores propõem uma arquitetura que preserva a privacidade de objetos inteligentes (*things*) através de controlo de acesso anónimo para uma plataforma IoT M2M.

Neste trabalho é usada uma tecnologia, muito embora tenham sido propostas diversas outras como alternativas. O Distributed Capability-based Access Control (DCapBAC) [16] é o mecanismo principal utilizado e idealizado/proposto pelos mesmos autores. Este mecanismo de controlo de acesso é baseado em *tokens* de autorização contendo privilégios de acesso que foram previamente garantidos a um utilizador. Baseado em outros mecanismos existentes, o DCapBAC permite implementações distribuídas no contexto do IoT. O *token* usa o formato JavaScript Object Notation (JSON) [10] que é anexado às mensagens de pedido de permissão de acesso usando o Constrained Application Protocol (CoAP) [61]. O DCapBAC associa privilégios a indivíduos identificados pela sua chave pública (de certificados X.509) sendo garantida a identificação inequívoca de cada utilizador. O DCapBAC é seguro e fornece um controlo de acesso flexível para o ambiente IoT, no entanto não possui mecanismos de garantia de privacidade no acesso. Para tal, os autores sugeriram uma modificação a este modelo. De forma a garantir privacidade no acesso, os autores usaram o conceito de pseudónimos e políticas associados a um utilizador ao invés dessa associação ser feita com uma chave pública. A sua utilização não está suficientemente detalhada neste trabalho. pelo que é difícil avaliar a sua utilização no contexto do IoT.

Para além deste mecanismo, foram propostos como alternativas muitos outros que permitem efetuar o controlo de acesso sobre os dados. Uma das alternativas é o Identity-Based Encryption (IBE). Este mecanismo de cifra consiste na cifra de dados sem que seja necessária uma chave pública gerada - a chave pública usada pode ser o identificador direto de uma entidade. Este mecanismo pode ser questionável, pois a existência da chave pública também poderia funcionar como um método de identificação. Para além disso, os métodos usados para a criação das chaves privadas pode não ser o mais seguro. Outra alternativa usada é o CP-ABE já explicado. A última alternativa é o Identity Mixer (Idemix). Esta alternativa é um sistema de credenciais anónimos que possibilita a divulgação seletiva de informação de identificação para garantir anonimato. Ao contrário dos mecanismos tradicionais de gestão de identidade (certificados X.509), os utilizadores podem obter credenciais para provar que satisfazem determinados atributos de identidade sem que informação adicional seja divulgada. Estes atributos são elementos criptográficos. O seu dono tem a possibilidade de selecionar quais atributos pertencentes à sua identidade podem ser revelados. O Idemix permite ainda a prova de posse de uma assinatura sem que nem a assinatura nem as mensagens inerentes à mesma necessitem de ser revelados, utilizando provas ZK.

No restante trabalho, foram indicadas formas de uso de cada uma destas tecnologias

de modo a que a privacidade no acesso a dados no contexto de IoT seja mantida para os diferentes mecanismos alternativos, que teriam de ser adaptados ao contexto de utilização para que se pudesse usufruir da prometida privacidade de acesso.

Este trabalho foca-se no acesso anónimo e privado o que não é o objetivo principal desta dissertação. Foi incluído nesta secção de forma a mencionar a existência de diversas tecnologias de controlo de acesso e cifra de dados com controlo de acesso embutido para conseguir um armazenamento seguro dos dados remotamente.

3.3 OUTROS

Foram ainda usados [44], [62] na busca de informação adicional acerca dos tópicos da segurança em IoT e/ou *cloud*. Os trabalhos contém uma análise sobre trabalhos relacionados com esta temática que foram já desenvolvidos. Para além destes foram consultadas algumas outras referências. No entanto, estas não serão aqui referidas, devido à baixa quantidade, repetição ou presença de informação irrelevante.

Conclui-se assim este capítulo onde foi analisado o estado da arte atual relativamente ao tema da segurança e privacidade de dados produzidos no ambiente da IoT. Como foi verificado, a área em questão carece de soluções unificadas que possam resolver o problema da privacidade dos donos dos dados evitando que a informação produzida chegue às mãos de terceiros.

CAPÍTULO 4

Problemas e Requisitos

Neste capítulo é feita uma primeira abordagem sobre os problemas que se pretendem resolver nesta dissertação.
Começa-se por mencionar e explicar problemas de segurança existentes nas plataformas IoT e os requisitos variados que a solução proposta no Capítulo 5 terá de cumprir.

4.1 requisitos de segurança na iot

Existem diversos problemas de segurança no âmbito da IoT, como foi referido anteriormente. Nesta secção serão descritos os principais problemas que serão resolvidos com a proposta de uma arquitetura solução para uma plataforma IoT. Estes problemas podem ser incluídos nas seguintes categorias:

- Privacidade dos dados - privacidade esta que tem de ser tratada a nível do armazenamento e da transferência pela rede ou entidades não confiáveis;
- Controlo de granularidade no acesso aos dados;
- Controlo/autorização de acesso aos dados por parte dos seus donos;
- Autenticação das diferentes entidades;
- Controlo de integridade dos dados;
- Distribuição de poder sobre tipos de dados por diversas entidades.

4.1.1 zonas confiáveis e não confiáveis

Normalmente, todas as arquiteturas que têm em vista tratar de problemas de segurança, têm de aplicar os diferentes mecanismos que existem de forma a combater a possibilidade e as vulnerabilidades a ataques de um determinado sistema. Nestas

arquiteturas existe, normalmente, uma zona ou um conjunto de dispositivos que são assumidos seguros uma vez que estão localizadas numa determinada propriedade ou estão sob o controlo de proprietários. Todos os dados que podem ser encontrados em claro ou dedutíveis fora dessas zonas ou dispositivos seguros são alvos de ataques por parte de terceiros pondo em risco a privacidade e a segurança dos utilizadores. Caso tal não se assumisse, seria impossível idealizar um sistema com um elevado nível de segurança (nunca se pode dizer que um sistema é 100% seguro, uma vez que nunca se sabe quando pode ser encontrada uma vulnerabilidade nos mecanismos de segurança que são aplicados ou pode ser alvo de um ataque por canal lateral), pois existirão sempre problemas de privacidade, isto é, os dados estariam sempre vulneráveis caso, por exemplo, um atacante conseguisse obter controlo sobre o contador inteligente que mede os níveis de consumo de gás de uma residência manipulando-o para obter a informação, supostamente privada, desejada.

Estas zonas/dispositivos seguros, no contexto do caso de uso dos contadores inteligentes poderiam, por exemplo, ser o próprio dispositivo produtor de dados: o contador. Assim, considera-se que este dispositivo é invulnerável a ataques no ponto de vista da arquitetura para uma plataforma IoT, para que se consiga obter segurança em todo o sistema. Alternativamente, pode-se definir uma zona segura de dados, isto é, um perímetro de segurança. Tal como no caso anterior, considera-se que todos os dispositivos e comunicações feitas dentro deste perímetro são invulneráveis a ataques no ponto de vista da arquitetura para uma plataforma IoT que pretende resolver um conjunto de problemas de segurança. No contexto dos medidores inteligentes, um perímetro de segurança é, por exemplo, a zona que contém o medidor em si, o canal de comunicação que existe e que o medidor usa para enviar dados para o *gateway* e o próprio *gateway*. Nos casos em que isto se aplica, é de notar que o *gateway* necessita de estar no domínio da propriedade do utilizador, caso contrário a suposição de segurança deste perímetro não poderia ser aceitável.

Para além das zonas/dispositivos proprietários que se podem assumir seguros, existem certas entidades, zonas, componentes da plataforma IoT ou dispositivos que podem ser assumidos como confiáveis. Estes já não necessitam de estar dentro da propriedade do produtor dos dados que necessitam de ser protegidos. Estes podem estar em qualquer localização na rede. A principal diferença entre componentes confiáveis e componentes proprietários seguros é que não se pode assumir que os primeiros possam obter acesso aos tipos de dados em claro necessários para que a finalidade da aplicação de segurança (a finalidade da privacidade dos dados é o foco desta dissertação) não se consiga atingir. Mesmo que estes componentes sejam confiáveis, continua a existir a necessidade de aplicar mecanismos de segurança com o objetivo de "esconder" e proteger informação ou então diminuir ao máximo o acesso aos diferentes tipos de informação

a que a entidade confiável tem acesso. Ou seja, a entidade confiável não pode ter acesso a tipos de dados suficientes que permitam o abuso de segurança. Para tal, é conveniente distribuir os diferentes tipos de dados a que cada entidade tem acesso de forma a diminuir a possibilidade da entidade confiável prejudicar a segurança de todo o sistema. É assumido que os componentes funcionam devidamente e cujo *software* e *hardware* não foi alterado de forma a efetuar processamentos maliciosos que podem, de alguma forma, obter acesso a dados ilegalmente. Assim, estes componentes são considerados seguros ao nível do seu funcionamento. No entanto, não podem possuir informação que, manipulada ou não, possa levar à revelação de dados privados/sensíveis. Denominam-se de TTPs e diz-se que este tipo de componentes são corretos (na medida em que funcionam e fazem o processamento corretamente), mas curiosos (na medida em que, a qualquer momento, a informação lá armazenada ou que por lá flui, pode ser modificada, acedida e interpretada). No entanto, existe sempre a possibilidade de duas entidades confiáveis diferentes praticarem conluio. Tal poderia levar a um grande problema de segurança. Problema esse que é sempre difícil de resolver, principalmente quando existem diversas ligações entre dados e estes necessitam de navegar pela rede e para fora de zonas seguras.

Como é natural, não se pode garantir que um componente confiável funcione sempre corretamente e não seja vulnerável a ataques de modificação de *software*/*hardware* e, por isso, não é desejável a sua existência.

4.1.2 PRIVACIDADE DOS DADOS

No contexto da IoT existe um grande problema a nível de segurança: a privacidade dos dados dos utilizadores. Este é um dos problemas de segurança que nos levou à elaboração desta dissertação e também ao desenvolvimento da arquitetura proposta.

Em arquiteturas IoT convencionais, a falta de privacidade dos dados de quem os produz pode ter diversas consequências graves. Imaginemos que um determinado utilizador de uma rede IoT produz dados e envia-os para a rede. A partir do momento em que os dados saem da zona "segura" do utilizador, isto é, o dispositivo onde são produzidos, é perdido todo o controlo que o dono (o utilizador) dos dados tinha sobre eles. Esta perda de controlo acontece porque, uma vez que os dados abandonam o dispositivo sem proteção, o utilizador fica sem a noção da localização dos dados, sem a noção do número de cópias dos dados já efetuadas e sem a noção do número de indivíduos/entidades que tenham os seus próprios dados em mão, ilegalmente e violando a privacidade do produtor da informação em questão. Assim que os dados deixam os produtores, qualquer outra entidade que possa vir a obter esta informação pode expo-la

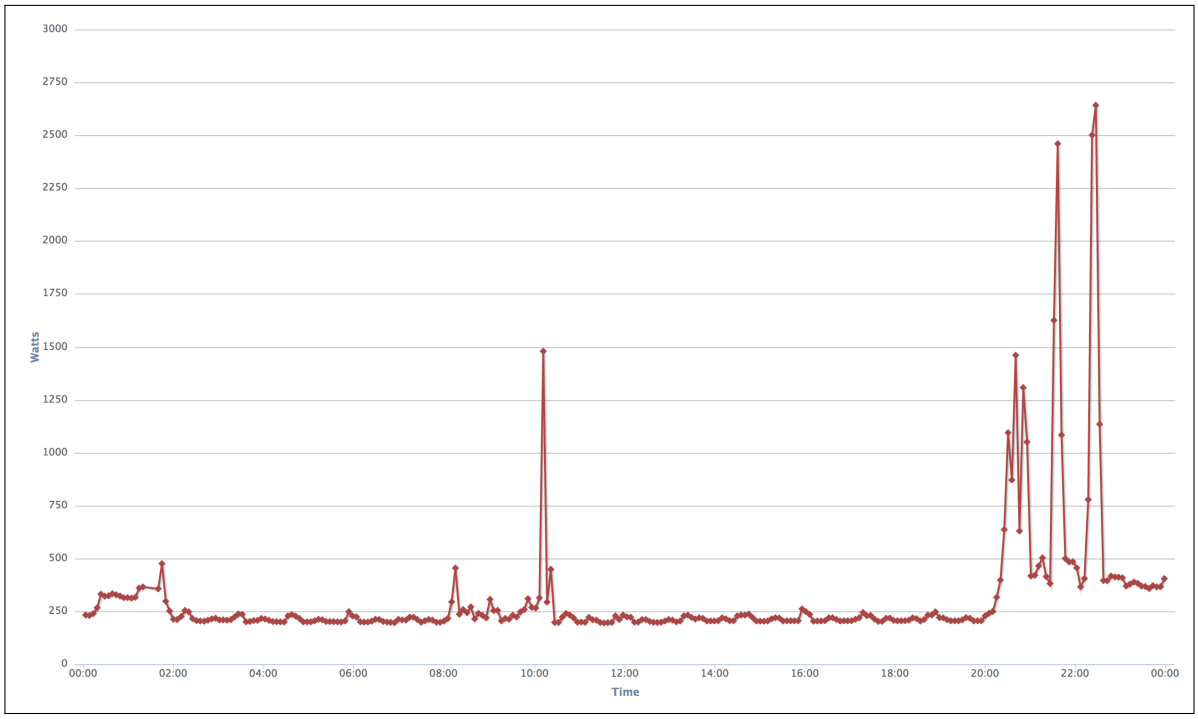

Figura 4.1: Gráfico da análise dos dados de um contador inteligente

publicamente; pode utilizá-la como informação sensível de credenciais para obter acesso ilegal a um determinado sistema, uma residência ou uma plataforma *software online*; pode vendê-la a entidades terceiras que, posteriormente, a podem utilizar para efeitos de publicidade não solicitada, pode utilizá-la para obter informações confidenciais sobre os utilizadores como seres humanos e prejudicá-los de forma física ou psicológica, de alguma maneira; pode comprometer a segurança de um país no caso de este fazer uso de uma plataforma IoT para produção e coleção de dados de diversas fontes; entre muitas outras forma de uso indevido de informação privada capturada por terceiros que não os donos desses mesmos dados.

Uma das aplicações da IoT é em residências ou edifícios que fazem uso de eletricidade, gás, água, etc. Para tal, são normalmente usados diversos contadores e medidores de quantidades. Produzem dados tendo em conta o contexto em que se incluem e para que foram concebidos. Estes dispositivos, quando associados a uma rede IoT e com as capacidades necessárias para que possam interagir com os outros componentes IoT, necessitam de enviar as informações produzidas para a rede. Esta informação é, na grande maioria das vezes, enviada primeiramente para um *gateway*. Após a receção dos dados, o *gateway*, procede então com o envio dos mesmos para a plataforma IoT que irá tratá-los, armazená-los e prepará-los para que possam posteriormente serem consultados por consumidores.

Neste contexto, é necessário tomar medidas para que os dados saiam da zona segura

de produção de dados do utilizador em questão (dono dos dados) devidamente protegidos. Caso contrário, podem surgir complicações ao nível da privacidade dos dados. Neste contexto, a divulgação de medições e contagens efetuadas por medidores/contadores de gás, eletricidade e água podem trazer graves consequências. Imaginemos que não existe qualquer mecanismo de segurança dos dados numa determinada plataforma IoT. Caso os dados vazem da zona de segurança e um atacante se apoderasse dos mesmos, ele poderia aceder a informação vital sobre os utilizadores. Como é possível observar, na Figura 4.1 está presente um gráfico originado com a ajuda dos dados recolhidos por um contador inteligente de eletricidade localizado numa residência habitada por uma família comum. O eixo horizontal contém os instantes de tempo de amostragem em horas e o eixo vertical contém os valores medidos de energia em *watts*. Os pontos vermelhos representam uma medição efetuada. Verifica-se, portanto, que o gráfico representa a energia consumida pela residência em questão no intervalo de um dia. Desde as duas horas até perto das oito horas da manhã deste dia, verificam-se utilizações de energia mais ou menos constantes. O atacante, através da análise destes dados e conhecendo padrões de utilização de energia de diversos eletrodomésticos e equipamentos (o que pode requerer um estudo adicional), pode afirmar, com uma grande probabilidade de acerto, que, durante a noite, esta residência apenas tem um consumidor de energia elétrica a trabalhar e ligado à eletricidade. Através destas inferências e com a ajuda de outros dados medidos em diferentes dias, o atacante pode verificar se é muito frequente ou não a existência de atividade noturna por parte dos habitantes desta residência. Caso tal comportamento seja frequente, o atacante pode planear um certo tipo de ataque físico que tenha a residência (ou até mesmo os seus habitantes) como alvo. Caso se verifique a existência de atividade durante a noite: por exemplo um aumento de energia consumida de três em três horas, o atacante pode, por exemplo, deduzir que existem bebés presentes na residência pois estes necessitam de ser amamentados nos períodos de tempo referidos e, para tal, é necessário ligar luzes e, eventualmente, aquecer leite num micro-ondas. Esta informação obtida por atacantes, que apenas analisam os dados recolhidos e não seguros de medições efetuadas por contadores inteligentes de plataformas IoT, permite aos primeiros obter informações sensíveis e privadas dos donos dos dados. Continuando a análise do gráfico da Figura 4.1, verifica-se um aumento e variações de quantidade de energia que está a ser utilizada entre as oito horas da manhã e por volta das dez horas e trinta minutos da manhã. A energia gasta corresponde, muito provavelmente, à quantidade de eletricidade necessária para que os aparelhos de banhos e eletrodomésticos para preparação de pequeno-almoço possam funcionar corretamente. O atacante sabe, assim, que é naquele intervalo de tempo que as pessoas se preparam para saírem de casa para o trabalho. Para além disso, é possível também determinar o número de pessoas que se levanta e toma o pequeno-almoço através da

análise dos padrões de energia. Durante a tarde, caso se verifique um consumo de energia em muito semelhante ao verificado durante a noite (como acontece neste caso), o atacante pode inferir que não está ninguém presente na habitação em um determinado dia da semana devido à frequência de padrões de baixa utilização da energia elétrica às quintas-feiras, por exemplo. Análises similares podem ser efetuadas aos restantes intervalos de tempo do dia em questão. Esta análise também pode ser aplicada em contadores de água e de gás. Medições de consumo de água pode acrescentar informação adicional que, combinada com medições de outros tipos ajudam a elaborar inferências sobre o estilo de vida dos residentes. Um exemplo é: um residente pode levantar-se a meio da noite para ir à casa de banho e utilizar apenas água, não sendo necessário um consumo de eletricidade muito significativo. Ainda neste caso, é possível verificar uma explosão na quantidade de energia que é gasta à noite, entre as vinte horas e as duas horas da manhã, podendo significar que uma placa de indução ou uma máquina de lavar a loiça estariam em funcionamento. Desta forma, é altamente provável que um atacante saiba que, naquele dia, os residentes jantaram em casa e não saíram de casa após a refeição.

Existe ainda uma outra questão pela qual não entraremos com muito detalhe. Essa questão é a lei existente para o armazenamento seguro e proteção de informação sensível em bases de dados exteriores [8], [13]. Existem normas legais acerca deste assunto. Caso dados sensíveis não estejam devidamente protegidos (ou seja, fortemente cifrados) aquando do seu armazenamento, tal pode constituir uma violação à lei em alguns países. A falta de privacidade dos donos deste tipos de dados não é tolerada e os responsáveis por plataformas deste género podem ser vir a ser sancionados judicialmente.

Como é possível verificar, a privacidade e o controlo dos dados que o utilizador produz pode trazer diversos e graves problemas, caso não seja devidamente respeitada e protegida. É a pensar na privacidade dos residentes que utilizam contadores e medidores inteligentes de eletricidade, água e gás que se propõe uma arquitetura IoT que preserva a segurança deste tipo de informação, mantendo a privacidade do utilizador intacta, sendo que estes apenas podem ser acedidos no domínio seguro e proprietário do utilizador. Contudo, a arquitetura que é apresentada também é válida para outros casos de uso de plataformas IoT que procurem manter a privacidade dos dados dos seus utilizadores.

4.1.3 CONTROLO/AUTORIZAÇÃO DE ACESSO AOS DADOS

Para além de questões de acesso indevido a dados privados/sensíveis através de ataques aos diversos componentes de uma plataforma IoT incluindo dispositivos, entidades, canais de comunicação e outros, é necessário ter em conta o acesso indevido a esta

informação através do uso correto, planeado e legal da plataforma. Este acesso pode ser feito a partir de um consumidor de dados que necessite de aceder a este tipo de informação produzida. No entanto a aplicação em questão pode, ou não, ser autorizada a ter acesso a ela. Caso todas as aplicações pudessem aceder a toda a informação que é armazenada pela plataforma IoT, então o problema de controlo de acesso aos dados não se põe, uma vez que não é necessário obter uma autorização para a obtenção de determinados dados por uma determinada aplicação. No entanto, em situações reais, estas aplicações consumidoras apenas podem obter informação acerca de um conjunto bem definido de dados produzidos por certos dispositivos.

Nas plataformas IoT reais, as aplicações/entidades são desenvolvidas por terceiros que podem ser maliciosos e podem ter uma intenção e um objetivo final diferente daquele que seria o correto e legal. Um possível comportamento malicioso pela aplicação tem de ser tratado. Para além desta ter de ser autenticada - o que leva a um outro problema - a aplicação necessita de ser autorizada. Caso uma aplicação consiga obter dados a que não é suposto ter acesso, tal pode resultar em problemas de privacidade e uso indevido dos mesmos. As consequências deste acontecimento são as mesmas que se verificam quando a privacidade dos dados não é respeitada. Estas consequências foram referidas na Secção 4.1.2.

A autorização não existe para que seja provada a identidade da entidade com quem se está a comunicar. O consumidor, a um determinado momento, necessita de aceder a um conjunto de informação. Para o fazer, constrói um pedido (*request*) de acesso aos dados de um determinado dispositivo. Este pedido flui até um certo componente ou entidade de autorização pertencente à plataforma IoT ou uma entidade confiável. Este componente procede então à autorização de acesso. Pode não ser necessário um componente ou entidade para efetuar esta operação, dependendo dos mecanismos de segurança utilizados, nomeadamente na aplicação do mecanismo de CP-ABE [9] ou KP-ABE [26], por exemplo. Estes mecanismos foram explicados na Secção 3.2.2 e são formas de controlo de acesso que utilizam metodologias novas ao invés das implementações tradicionais de controlo de acesso como o DAC, o MAC ou o RBAC. Assim que o pedido chega ao componente de autorização mencionado, o processo de autorização procede-se e são consultadas políticas ou listas de controlo de acesso para que se decida finalmente se o acesso pode acontecer ou não. A resposta é, então enviada de volta à aplicação com os dados necessários caso tenha sido aceite o acesso ou com uma resposta negativa, caso contrário. Outra forma de efetuar a autorização, ao invés da configuração por parte do utilizador de ACLs ou outro elemento do modelo de controlo de acesso utilizado, será a de que requisitar, em tempo real, a intervenção do dono dos dados para que este permita ou não, de forma expressa, que os seus dados sejam acedidos por determinada aplicação/entidade em um determinado instante de tempo

de acordo com determinadas condições. Quando um pedido chega ao componente, é adicionado a uma lista de espera onde o utilizador será capaz de aceder e autorizar ou não de modo físico e interativo com o componente de autorização. Ao dar esta possibilidade ao utilizador, torna-se possível fazer com que se sinta em total controlo sobre os seus dados e que apenas após a sua revisão de um pedido feito por uma entidade consumidora, é que esta pode ou não obter uma vista transformada sobre a sua informação.

Ao efetuar a autorização de acesso aos dados, está-se a proporcionar a funcionalidade de segurança de controlo de acesso.

4.1.4 CONTROLO DE GRANULARIDADE NO ACESSO

Existe ainda um problema que se pretende resolver nesta dissertação e que está diretamente ligada à privacidade dos dados. Este problema é a granularidade dos dados que são mostrados às aplicações que os pedem.

Numa arquitetura IoT convencional, no momento da elaboração de um pedido de acesso aos dados construído pela aplicação e após este pedido ter sido aceite pelo mecanismo de autorização implementado na plataforma - imaginando que esta deve poder aceder aos dados com toda a legitimidade - os dados chegarão ao consumidor, de alguma forma. Uma vez que não existe qualquer preocupação com a segurança dos dados e com a granularidade com que esta é armazenada pela plataforma IoT e apresentada à aplicação, a informação que a última terá a oportunidade de apresentar aos seus utilizadores será a informação tal e qual esta foi armazenada. Como os dados que foram armazenados (ignorando que tenham sido, eventualmente, aplicados mecanismos de cifra sobre eles) são os mesmos dados que foram produzidos e que abandonaram o seu dispositivo produtor, concluímos que os dados que são apresentados aos utilizadores das aplicações são exatamente os mesmos que foram originados pelos dispositivos produtores localizados na camada mais baixa de uma plataforma IoT.

Dado que os consumidores têm acesso aos dados originados pelos produtores na sua íntegra, ter-se-ia que estabelecer uma relação forte de confiança entre os utilizadores donos dos dispositivos produtores que fornecem os dados e as aplicações consumidoras que os iriam obter. Ou seja, caso não se aplique nenhum mecanismo de segurança que limite o nível de granularidade dos dados que o consumidor pode aceder, este necessita de ser fortemente confiável. Esta relação de confiança pode não ser suficiente para que se consiga manter os dados em questão privados. Não é uma relação de confiança estabelecida que irá impedir as pessoas, empresa ou entidades utilizadoras da aplicação consumidora de vazarem os dados, venderem-nos, utilizá-los ilegalmente

ou estudá-los de forma a inferir informações que podem ser sensíveis e deveriam ter-se mantido privadas para evitar o risco de inúmeros tipos de ataques, já anteriormente referidos. Por isso, esta relação não deverá existir. Para tal, e de forma a manter a privacidade dos dados, é necessário aplicar mecanismos de proteção contra a revelação dos dados produzidos na sua íntegra, introduzindo alguma generalidade aos mesmos. Um exemplo de generalização é a aplicação do k-anonimato mencionado anteriormente que é um mecanismo de anonimato de dados que os generaliza de forma a que não seja possível obter informações demasiado detalhadas sobre o originador de dados, mas de forma a que seja possível retirar destes informação suficientemente precisa e relevante de maneira a que torne a aplicação que consome os dados útil e desempenhe as funções com uma determinada precisão para que foi concebida.

É preciso, assim, encontrar o equilíbrio entre o nível de utilidade que os dados transformados têm para o consumidor e o nível de privacidade dos dados que é conferido ao produtor.

Uma generalização executada sobre os dados pode ser feita de diferentes formas conforme o nível de privacidade/utilidade que se pretende obter no final. Caso a generalização seja demasiado leve, ou seja, a granularidade dos dados que seriam mostrados ao consumidor final seja demasiado baixa, o último teria a oportunidade de processar, apresentar ou fazer quaisquer outras ações sobre os dados de forma precisa, relevante e realmente útil. Todo este tratamento seria executado como se os dados em questão estivessem muito próximos dos dados verdadeiros, não generalizados/transformados. Esta questão traz um grande problema. Os consumidores, ao terem acesso a informação com um nível de granularidade tão baixo, resultaria num nível de segurança e privacidade da informação muito baixo, uma vez que os consumidores poderiam deduzir muito facilmente os dados reais dada a extrema aproximação da generalização com o detalhe real originalmente produzido. Tal falta de privacidade origina graves problemas, como foi discutido anteriormente. Como exemplo, imaginemos que um dispositivo produtor de dados tira uma fotografia e essa fotografia, antes de ser enviada para a rede IoT, é desfocada ligeiramente, ou seja, foi aplicado um filtro de desfocagem muito leve na imagem que quase que permite a sua visualização detalhada. Após esta transformação, a imagem é enviada (cifrada) para ser armazenada pela plataforma. Assim que a imagem chega até a uma aplicação de visualização de imagens (consumidora), esta é decifrada e mostrada desfocada. Como o nível de desfocagem é demasiado leve, é possível ver ou deduzir o seu conteúdo, tornando a aplicação útil, uma vez que mostra a imagem com um certo nível de detalhe, suficiente para o propósito final (por exemplo, verificação do nível de luminosidade do local onde foi tirada a fotografia). Contudo, o elevado nível de utilidade vem com um preço. O preço é o baixo nível de privacidade. Devido à grande quantidade de detalhe apresentado, é possível através da aplicação retirar mais

informações (úteis a atacantes ou outras entidades) sobre a imagem, como por exemplo, as cores de carros que estejam estacionados no referido local.

Caso a generalização seja demasiado forte, ou seja, a granularidade dos dados que seriam mostrados ao consumidor final seja demasiado elevada, o último não conseguiria processar, apresentar ou fazer quaisquer outras ações sobre os dados de forma tão precisa, tão relevante e tão útil como no caso anterior. Todo este tratamento seria executado como se os dados em questão estivessem muito afastados dos dados verdadeiros, não generalizados/transformados. Esta questão traz um grande benefício, relativamente ao caso anterior. Os consumidores, ao terem acesso a informação com um nível de granularidade tão elevado, resultaria num nível de privacidade da informação muito alto, uma vez que os consumidores não poderiam deduzir com facilidade os dados reais dado o extremo afastamento da generalização com o detalhe real originalmente produzido. Recorreremos ao mesmo exemplo que antes, o exemplo da fotografia, sendo que, desta vez, a fotografia, em vez de ser desfocada ligeiramente, será altamente desfocada sem que quase seja possível deduzir o seu conteúdo. Foi, então, aplicado um filtro de desfocagem muito pesado na imagem. Assim que a imagem chega até a uma aplicação de visualização de imagens (consumidora), esta é decifrada e mostrada desfocada. Como o nível de desfocagem é demasiado alto, não é possível ver ou deduzir o seu conteúdo, tornando a aplicação muito mais inútil e podendo, até, ser insuficiente para atingir o propósito final original de conceção da aplicação, uma vez que esconde todo, ou grande parte, do detalhe da imagem. Contudo, o diminuto nível de utilidade vem com uma grande vantagem. Ela é o alto nível de privacidade da imagem. Devido à baixa quantidade de detalhe apresentado, não é possível através da aplicação retirar muitas mais informações (úteis a atacantes ou outras entidades) sobre a imagem, como por exemplo, as cores de carros que estejam estacionados num determinado local.

A conclusão a retirar é que é necessário encontrar um mecanismo que esconda suficientemente bem os dados para garantir a privacidade dos produtores mas que sejam suficientemente úteis de forma a serem úteis para os consumidores.

A analogia feita acima com a imagem, pode ser aplicada a conjuntos de valores medidos por medidores de água inteligentes de uma residência, por exemplo. Onde o produtor dos dados é o medidor inteligente e o consumidor é uma aplicação desenvolvida por uma empresa que pretende estudar padrões de consumo de água sobre um conjunto de residências no país.

Nesta dissertação, também se procura resolver o problema da divulgação de informação privada na sua íntegra às diferentes aplicações, mesmo aquelas cujo acesso a essa informação seja permitida (autorizada). Isto porque não é possível manter a privacidade e a segurança dos dados de um utilizador caso estes sejam acedidos por outros consumidores, por mais confiáveis e corretos que estes sejam. A confiança não

impede que os consumidores abusem da informação e nada os impede de executar procedimentos maliciosos/ilegais com eles, pois os donos originadores da informação podem nunca vir a saber de tal, perdendo o controlo total sobre informação que pode ser sensível para os utilizadores.

4.1.5 AUTENTICAÇÃO

A falta de autenticação de todos os componentes é um problema grave de segurança. Não entraremos em muito detalhe neste problema de segurança, uma vez que o impacto da não resolução deste problema no ambiente IoT é semelhante ao impacto em qualquer outro sistema. A ausência de uma prova por parte de quaisquer duas entidades que comuniquem entre si geram uma dúvida em cada componente interveniente: "será que estou realmente a falar com a entidade que penso que estou a falar?". Assim sendo, é necessário a existência de autenticação entre as entidades de forma a assegurar a segurança dos dados no sistema. Para além disso, os componentes presentes na zona segura do proprietário dos dados, necessitam de se autenticar perante a plataforma IoT e outras entidades com que venha a comunicar utilizando a plataforma como um meio de comunicação (autenticação *end-to-end*). O utilizador deverá também estar ao corrente do estado da autenticação entre os seus dispositivos e outras eventuais entidades. Para além da autenticação entre estes componentes, também é vital que a plataforma (e as outras entidades com quem comunica) saiba realmente qual a aplicação consumidora que está a tentar obter dados. Esta aplicação deve-se autenticar de forma a que todo o restante sistema saiba qual entidade está a efetuar um pedido de acesso a dados. Ao ser efetuada esta autenticação, o utilizador deverá ter na sua mão a decisão de autorização ou não do acesso aos seus dados.

Assim, a apresentação de provas entre as várias entidades ou evidências físicas, que o utilizador possa ter acesso e visualizar, pode ser muito útil, nomeadamente nos casos em que o utilizador está a fornecer informação pessoal confidencial, onde cada troca de informação pode ser crucial na manutenção da sua privacidade.

4.1.6 CONTROLO DE INTEGRIDADE

O controlo de integridade dos dados é um requisito de segurança essencial. Toda a informação que é transmitida diretamente entre componentes necessita de ter um controlo de integridade de forma a evitar que terceiros modifiquem e corrompam a informação que está a passar no canal de comunicação. No caso do controlo de integridade efetuado ponto-a-ponto, uma vez que as arquiteturas de segurança necessitam

de ter em conta a existência de ataques *man-in-the-middle*, a verificação de integridade das mensagens acaba por não ser muito importante. O controlo de integridade das mensagens por si só não adianta muito na conceção de sistemas seguros, pois o referido ataque pode quebrar a implementação de mecanismos de controlo de segurança e torná-los inúteis. Tipicamente o controlo de integridade alia-se a mecanismos de autenticação, como a aplicação da infraestrutura de chaves públicas (Public Key Infrastructure (PKI)) através do uso de assinaturas, que já garantem a integridade dos dados ponto-a-ponto e, adicionalmente, é validada a autenticação de ambas as partes que participam na comunicação. Desta forma é também possível evitar ataques *man-in-the-middle*.

Para além do controlo de integridade ponto-a-ponto de todas as mensagens que fluem de uma entidade para outra, tem de existir ainda um controlo de integridade dos dados produzidos extremo-a-extremo. No momento de produção dos dados, estes terão de fluir para a plataforma onde ficarão armazenados de forma persistente. A partir daí, seguem para um determinado consumidor quando solicitados. Neste caso, é importante garantir que o consumidor dos dados recebe os dados corretos e não dados que foram corrompidos ou modificados propositadamente enquanto estavam armazenados. Caso tal acontecesse, não seria possível garantir que o consumidor recebia os dados corretos, mesmo que estes estivessem sujeitos a um determinado mecanismo de cifra. A não receção de dados corretos e a falta de integridade e fiabilidade nos mesmos tornaria o sistema IoT inútil.

4.1.7 DISTRIBUIÇÃO DE PODER

Um outro problema existente no ambiente do IoT é que, atualmente e em plataformas IoT convencionais e sem mecanismos de segurança implementados, a plataforma tem acesso aos dados em claro originários das *gateways* ou dispositivos produtores. Ora, de forma a resolver este problema, os dados terão de cifrados antes de chegarem à plataforma. De forma a que estes dados sejam cifrados, terão de existir chaves criptográficas simétricas. Assim, terá de existir um componente ou entidade que irá tratar de gerir estas chaves e certificar-se que as mesmas estão associadas a um utilizador ou cliente da plataforma IoT.

No processo de elaboração de uma arquitetura segura, estes três elementos, dados cifrados, chaves criptográficas e identificação de utilizadores, necessitarão de estar armazenados e presentes em algum local no sistema. O problema é que, excluindo o domínio do utilizador (dispositivos produtores e/ou *gateways*), uma vez que é considerado que este domínio é seguro (assumindo, pelo menos, que os meios de comunicação utilizados entre estes componentes é seguro), qualquer outra entidade não pode ter

acesso a todos estes três tipos de informação combinados. Se tal acontecesse, o problema da privacidade dos dados do utilizador (Secção 4.1.2) continuaria a manter-se, uma vez que a entidade em questão poderia associar e conectar todas as "peças" e passar a ter acesso a informação que é considerada privada para os utilizadores donos dos dados cifrados em questão.

As entidades que irão albergar os tipos de dados mencionados acima não podem ter em sua posse a combinação completa dos três tipos de dados. Isto é, no máximo, apenas dois dos tipos de dados podem estar presentes na mesma entidade sem que o nível de segurança do sistema diminua consideravelmente. Como exemplo, se uma entidade tiver em sua posse os dados cifrados, pode também ter em sua posse as chaves de cifra, desde que não tenha acesso à informação do utilizador que produziu os dados em questão. Se tal acontecer, essa entidade poderá decifrar os dados e terá acesso a informação. Eventualmente essa informação será inútil (do ponto de vista que não viola a privacidade do utilizador em questão) uma vez que não é possível efetuar a ligação entre os dados em claro e um determinado utilizador. No entanto, e de forma a manter ao máximo a privacidade dos utilizadores especialmente em casos de uso de plataformas IoT em que os próprios dados podem dar indicações sobre o utilizador em questão (por exemplo, localizações com coordenadas GPS, números pessoais de determinadas contas pertencentes a um grupo demasiadamente pequeno de utilizadores, etc.), uma entidade que tenha acesso a este tipo de informação deve ser confiável ou pertencer a uma zona de segurança em que apenas o utilizador tem acesso. No caso em que uma entidade tem em sua posse estes dois tipos de dados, um outro tipo de dados novo é criado: os dados em claro, sendo que possuir os dados cifrados e as respetivas chaves criptográficas ou apenas possuir os dados em claro, são casos semelhantes.

Adicionalmente, uma determinada entidade pode ter em sua posse os dados cifrados e a identificação de utilizadores desde que não possua meios para decifra desses dados (chaves criptográficas). Tal entidade, apenas possuindo estes conjunto de dados, nunca poderá obter a informação tão desejada que um atacante ou violador de privacidade deseja, pois nunca conseguirá decifrar os dados a não ser que seja efetuado um trabalho de encontro de vulnerabilidades sobre o(s) algoritmo(s) e modo(s) de cifra utilizados no mencionado conjunto de dados cifrados.

De forma a passar pelas principais combinações dos tipos de dados mencionados, ainda existe a possibilidade de uma entidade ter em sua posse as chaves criptográficas juntamente com a informação do utilizador. Como é facilmente percebível, apenas com estes tipos de dados, a entidade não poderá obter acesso aos dados violando a privacidade do utilizador. No entanto, não é correto assumir que uma entidade nunca terá acesso aos dados cifrados, uma vez que estes dados fluem pela rede e podem ser facilmente "apanhados" por atacantes. Deste modo, a atribuição do poder de posse

de chaves associadas à cifra da informação que um determinado utilizador identificado produziu, deve ser evitado. No entanto, caso não haja alternativa, a entidade em questão deverá ser confiável ou protegida numa zona segura (por exemplo um equipamento que pertença ao utilizador). Ou seja, a mesma funciona e age corretamente (Secção 4.1.1).

Assim, de forma a que se possa fornecer garantias de segurança, é necessário distribuir o acesso de todos estes dados por diferentes entidades. Para tal, e pensando nas limitações que são impostas a cada componente, é vital para um sistema seguro que este problema seja resolvido.

A distribuição destes tipos de dados por diferentes entidades pode tornar-se complexo. Isto porque existem componentes/entidades que apresentam certas limitações de capacidade de processamento ou de ligação com a rede, como será explicado na Secção 4.2.2.

O desafio aqui está em como distribuir o poder que se obtém por apenas possuir estes tipos de dados pelas diferentes entidades/componentes, adicionando, efetivamente, segurança ao sistema, resolvendo os problemas que se têm vindo a mencionar nesta secção e evitar, ao máximo, não respeitar os requisitos de sistema que serão mencionados na Secção 4.2.

A distribuição de tipos de dados por diferentes entidades não garante a segurança do sistema, tal como nenhum outro método de segurança aplicado o garante. Isto porque são, frequentemente, encontradas vulnerabilidades em sistemas, algoritmos, chaves e em praticamente tudo o que se relaciona com segurança sendo impossível fazer com que um sistema, de que tipo for, seja 100% seguro. No entanto, no que toca ao assunto da distribuição de papéis por diferentes entidades, existe uma maior probabilidade da segurança ser quebrada devido à possibilidade de existência de conluio entre várias entidades que, quando combinadas, possuam os três tipos de dados em conjunto. Isto é, uma entidade que tenha em sua posse os dados cifrados e uma outra entidade que tenha em sua posse a informação do utilizador associada às chaves criptográficas dos dados cifrados mencionados, podem comunicar entre si utilizando canais de comunicação terceiros ao sistema e assim partilharem este tipo de informações. Isto pode acontecer em todos os sistemas, pelo que é difícil conseguir impedir que tal aconteça num sistema seguro.

4.2 OUTROS REQUISITOS E LIMITAÇÕES DO SISTEMA

A arquitetura que se propõe nesta dissertação foi desenhada com base em vários requisitos.

Sendo que a arquitetura proposta acrescenta mecanismos de segurança relativamente

a uma arquitetura de uma plataforma IoT convencional, os requisitos da arquitetura que descrevo nesta dissertação incluem os requisitos de um sistema/plataforma IoT convencional e que já existe hoje em dia. No entanto, nem todos estes requisitos podem ser aplicados. Para que seja possível incluir mecanismos de segurança, é necessário efetuar algumas alterações e até colocar algumas limitações. Não serão aqui detalhados todos os requisitos que uma plataforma IoT deve ter, no entanto, falaremos sobre as principais funcionalidades e requisitos não funcionais que esta deve ter acrescentando as exigências e atributos relacionados com a segurança que se propõe adicionar.

Uma plataforma IoT tem como o principal objetivo - como ilustrado na Figura 2.1 - permitir que dispositivos limitados em termos de processamento e memória - as coisas (*things*) - estejam ligados numa rede comum e com a restante rede pública - a Internet. Para efetuar esta ligação é necessário um sistema e uma plataforma que o suporte. Toda esta ligação entre estes componentes é chamada da "Internet das Coisas" (IoT).

Segundo o modelo de referência para uma plataforma IoT (Secção 2.1), todos estes sistemas necessitam de componentes como um produtor de dados, uma plataforma subjacente que comunique com estes e um consumidor de dados que os obtém a partir da plataforma.

De notar que, adicionalmente ao que irá ser dito nesta secção, assim como em adição aos requisitos principais que qualquer plataforma IoT deve proporcionar, o principal objetivo em que esta dissertação se foca, são os problemas de segurança das plataformas IoT que foram mencionados várias vezes ao longo desta dissertação, assim como na Secção 4.1. A sua resolução é o principal requisito do sistema.

Serão descritos requisitos de sistema adicionais e um tanto relevantes nas secções seguintes.

4.2.1 DISPOSITIVOS PRODUTORES

Os dispositivos produtores são essenciais numa plataforma IoT. Estas são as "coisas" na denominada "Internet das Coisas". Estes dispositivos necessitam de gerar informação tendo em conta o meio em que se encontram. Esta informação produzida terá de ser comunicada com o restantes componentes da IoT. Se tal não fosse possível, o principal propósito da IoT não seria cumprido. Contudo a partilha destes dados com o exterior necessita de ser condicionada. Um requisito que foi estabelecido na elaboração da arquitetura proposta foi o facto de estes dados apenas poderem ser acedidos na sua íntegra pelos próprios donos, seus criadores.

Como já referido anteriormente, nesta arquitetura é assumido que existe um perímetro de segurança (uma zona segura) e que pertence ao domínio do proprietário do

dispositivo produtor em questão. Esta zona segura contém os dispositivos produtores e o *gateway*, sendo que todas as comunicações existentes entre estes dois componentes encontra-se dentro do domínio proprietário do utilizador e, portanto, é segura.

Deste modo, os dados não podem, de maneira nenhuma, serem acedidos por outros que não os criadores, na sua íntegra. Contudo, uma transformação, uma vista ou uma generalização, como se queira chamar, dos dados produzidos já pode ser mostrada a terceiros. Ainda assim, estes terceiros têm de ser selecionados e necessitam de ter autorização para tal. Com este requisito de segurança abrangente e juntamente com a solução de arquitetura que é proposta pretende-se resolver os problemas de segurança enumerados e descritos na Secção 4.1.

Como já foi mencionado ao longo desta dissertação, os dispositivos produtores necessitam de enviar os seus dados para um *gateway* de modo a que os diferentes mecanismos de segurança sejam aplicados decentemente. Todos os dados que naveguem diretamente desde o produtor de dados para a plataforma IoT passando por uma rede insegura e não fazendo uso do *gateway* que garante a segurança dos dados, estão sujeitos a ataques de privacidade e ao roubo ou cópia dos dados por parte de terceiros - quer aqueles que são atacantes e que atuam invasoramente, quer os que não são atacantes mas têm os dados na sua posse (por exemplo a entidade cuja finalidade é o armazenamento de toda a informação produzida pelos dispositivos produtores). Assim, um dos requisitos da arquitetura proposta, limitativa dos requisitos que normalmente existem planeados para uma plataforma IoT convencional, é o facto de a comunicação entre os dispositivos produtores de dados com a rede/plataforma IoT ter de ser feita contando com um intermediário (um *proxy*). Na Secção 2.4, mais precisamente na Figura 2.2, que corresponde à descrição da arquitetura M2M elaborada pela ETSI, foi referido que, na ótica dos seus autores, existia a possibilidade de permissão de ligação de Devices M2M (os dispositivos produtores) diretamente à rede IoT sem que a informação passasse pelo Gateway M2M (*gateway*). No entanto, os dados, cuja privacidade deve ser mantida, devem sempre navegar desde o produtor para o *gateway*, este que aplica diversos mecanismos de segurança aos dados, seguindo para a rede localizada na camada acima e que comunica com a plataforma IoT. Desta forma, os dispositivos produtores nunca poderão estar diretamente ligados à plataforma IoT, exceto no caso de estes possuírem capacidades de processamento, memória e ligação de rede iguais ou muito semelhantes às capacidades de um *gateway* no estado da arte atual.

4.2.2 O *gateway*

O *gateway* também possui determinados requisitos no âmbito da IoT. Os mesmos são aplicados neste caso. O *gateway* tem como principal função agregar os dados de diversos sensores, por exemplo, e enviá-los para a plataforma. Deste modo, o *gateway* terá de ter capacidades de processamento superiores aos restantes dispositivos produtores de dados e ter uma capacidade de envio de dados para o exterior (plataforma) de forma rápida. Requisitos ao nível da escalabilidade para poder receber os dados de diversos dispositivos e requisitos de disponibilidade para garantir que os dados recolhidos são, efetivamente, enviados, são também vitais para o bom funcionamento do sistema.

Existe uma limitação neste componente. Uma vez que o *gateway* lida com dados sensíveis e que estão sob proteção, este componente necessita de estar localizado fisicamente em um local seguro. Este local passa por ser um local privado, de acesso condicionado a apenas pessoas autorizadas. Como, nesta dissertação, é baseada no caso de uso de medidores de energia elétrica inteligentes que pertencem a uma plataforma IoT, o local seguro, privado e restrito a que é referido é propriedade privada ou residência dos utilizadores cujos dados de consumo elétrico são monitorizados e contados por estes contadores inteligentes que produzem os dados privados referentes a esta residência. Deste modo, juntamente com os dispositivos produtores e todo o meio de comunicação entre eles, é formada uma zona segura ou um perímetro de segurança à volta destes componentes. A existência desta zona com estes componentes é obrigatória para garantir a privacidade dos dados.

Os *gateways* nem sempre são utilizados no contexto de residência segura e, por isso, são desenvolvidos de forma a poderem ser utilizados de forma mais abrangente. Assim, dada a sua posição geográfica muitas vezes adversa, tipicamente os *gateways* utilizam a rede móvel para efetuar esse tipo de comunicação com a plataforma IoT. A rede móvel é extremamente limitada, o que produz um requisito importante na elaboração de uma arquitetura segura, que são os seguintes:

- A quantidade de informação que abandona o *gateway* com destino a plataforma utilizando a rede móvel, necessita de ser reduzida. Isto porque a utilização de dados móveis é cara e, portanto, não é possível que largas quantidades de dados sejam trocados entre o *gateway* e a plataforma. Os próprios valores medidos e produzidos já têm um grande impacto no que toca ao tamanho dos dados enviados e, por isso, aumento dos *payloads* deste tipo de dados é de evitar, assim como a adição de novos pedidos e comunicações com a plataforma ou outras entidades que exijam trocas de informação muito acentuadas;
- A informação que chega ao *gateway* do exterior (plataforma ou outras entidades) deve ser extremamente reduzida. A frequência de receção de informação que

provém da rede deve ser o mais pequena possível. Muitos operadores de rede limitam extremamente o canal de comunicação no sentido inverso. Existem operadores que servem este tipo de componentes IoT que não permitem que tais comunicações sejam efetuadas mais que três ou quatro vezes por mês.

O que foi mencionado leva-nos a um outro principal requisito. A plataforma deve suportar que se estabeleçam ligações bidirecionais. Ou seja, a plataforma deverá ser capaz de enviar dados, ainda que limitados, para o dispositivo *gateway* (ou o dispositivo que estará a comunicar com a plataforma no envio de dados) e vice-versa. Este requisito funciona como uma condição da plataforma para que a arquitetura segura que é proposta nesta dissertação se possa concretizar.

Adicionalmente, para que a arquitetura segura que é proposta funcione de forma correta, é necessário que os dispositivos *gateways* tenham alguma forma de estabelecer uma comunicação unidirecional com o utilizador. Este último necessitará de visualizar um simples código de 4 caracteres, pelo que mostradores de segmentos de Organic Light Emitting Diodes (OLEDs) podem ser suficientes.

4.2.3 ARMAZENAMENTO DOS DADOS

Os dados provenientes do domínio privado, cifrados e protegidos (ou não) contra o acesso por parte de atacantes e por parte das entidades que estão em contacto com os dados, necessitam de ser armazenados persistentemente para que possam ser acedidos mais tarde.

No armazenamento dos dados, para além do correto funcionamento e de um armazenamento de dados fidedigno sem corrupção dos mesmos, o requisito mais importante será o deste componente permitir que a ação de inserção de dados no mesmo seja permitida de forma flexível. Isto é, este componente deve aceitar que qualquer tipo de dados seja lá armazenado. Isto porque, uma vez que os dados armazenados estarão cifrados, estes não farão sentido para o componente que trata da persistência pois não serão objetos estruturados (como JSON ou eXtensible Markup Language (XML)), daí ser necessário tal requisito.

De forma similar, um outro requisito importante é o deste componente permitir que a ação de seleção/acesso aos dados seja permitida de forma também flexível e simples. Isto é, este componente deve permitir que certos parâmetros sejam usados de forma a filtrar os dados e obter aqueles que são interessantes no momento de acesso. Para além disso, deve incluir funcionalidades de paginação no acesso aos dados. Como estamos a lidar com dados de grandes dimensões, uma grande quantidade de dados pode ser recuperada quando um pedido é efetuado a este componente. Assim, é necessário que

este possua capacidades de subdivisão dos mesmos em páginas e forneça mecanismos simples de acesso e de navegação por essas páginas. Na plataforma IoT que foi utilizada como ponto de partida para a elaboração da arquitetura de uma plataforma IoT segura, tal já foi tido em consideração.

4.2.4 DESEMPENHO E DISPONIBILIDADE

Qualquer plataforma IoT necessita de ter requisitos de desempenho. Só alcançando alto desempenho é que a plataforma consegue manipular os dados que são produzidos e enviados pelos dispositivos produtores em tempo real e com uma periodicidade de envio elevada. A capacidade que uma plataforma tem para o conseguir é uma característica fundamental e importante na avaliação global de uma plataforma IoT. Para além de não ser possível a comunicação de dados em tempo real, uma plataforma cujo desempenho deixa muito a desejar torna-se inutilizável. Podemos imaginar inúmeras situações onde a demora na execução de operações em ambiente IoT pode afetar a usabilidade. Por exemplo, o contexto dos medidores de energia elétrica inteligentes localizados em residências em que existem aplicações/entidades consumidoras que pretendam obter os dados medidos em tempo real. Caso os medidores estejam configurados e sejam requisitados a terem de enviar informação elétrica recolhida com uma periodicidade relativamente elevada (por exemplo de segundo em segundo) e na eventualidade da plataforma necessitar ainda de efetuar processamento sobre os dados cuja demora seja grande, as aplicações/entidades consumidoras não poderão atingir o seu objetivo da recolha de dados em tempo real. Mais um exemplo: o acesso aos dados ainda no mesmo caso de uso que foi mencionado. Se o acesso aos dados for demasiado complexo e necessite de inúmeros passos e comunicações entre muitos componentes, tal irá prejudicar o desempenho de obtenção de dados. Se uma aplicação consumidora tiver de esperar 2 minutos para obter a informação que pretende, a aplicação cairá rapidamente em desuso devido à falta de paciência da generalidade dos utilizadores ou entidades que a usam. Estes que pretendem ter sempre a informação à mão e sempre acessível sem impedimentos relacionados com o desempenho. Um último exemplo relacionado agora com um outro caso de uso: na domótica. Imaginemos que numa residência, um utilizador instalou uma televisão inteligente que pode ser comandada através de uma aplicação no contexto da IoT. Nenhum utilizador faria uso dessa aplicação caso esta demorasse 10 segundos para proceder à mudança de canal ou até mesmo ao aumento e diminuição do volume da TV.

Como é natural, os dados produzidos pelos dispositivos da camada mais baixa têm de chegar até à plataforma IoT e, após todo o processamento necessário, manter os

dados disponíveis. A informação deve estar acessível a qualquer momento, pois qualquer consumidor, desde que autorizado, poderá precisar do acesso. Um dos requisitos de uma plataforma IoT é a alta percentagem de disponibilidade dos dados que são recolhidos para que o sistema seja útil.

Tendo tudo isto em conta, o desempenho e disponibilidade, para além da segurança, fiabilidade e funcionalidade, são também muito importantes e podem ditar a preferência pelos utilizadores num sistema ou noutro, mais rápido e disponível, descartando certos mecanismos de segurança que o sistema mais lento proporcionava.

4.2.5 ESCALABILIDADE

A escalabilidade é um requisito extremamente importante na IoT. A não existência de escalabilidade num sistema deste tipo implica que a adição de novos serviços e produtos IoT não pode ser realizável de modo fácil, simples e rápido. A nova adição de serviços e produtos é essencial para que a plataforma IoT seja útil e não seja necessário efetuar a instalação de uma nova plataforma de forma separada para utilização por parte de cada serviço ou produto.

A adição de novos equipamentos produtores como sensores, dispositivos que colecionam dados do seu ambiente ou *gateways* deve ser possível ser efetuada com facilidade e simplicidade através de certos mecanismos de registo.

Quando se menciona a escalabilidade, a facilidade e simplicidade na adição de todos estes componentes e entidades à plataforma não são o único problema existente. Em um determinado sistema pode ser simples de atingir tal objetivo, no entanto, a carga da adição de todas essas entidades e equipamentos que, como consequência, adicionam a todo um sistema IoT pode ser de tal ordem que o próprio sistema deixe de funcionar corretamente ou alguns requisitos não funcionais como o desempenho ou disponibilidade não se verificarem a longo prazo de operação do sistema.

De notar que as arquiteturas de plataformas IoT têm já estes requisitos em mente. Um grande desafio na aplicação de segurança nessas plataformas, está em conseguir manter estes requisitos de desempenho, disponibilidade, escalabilidade e outros em um nível razoável e, ao mesmo tempo, garantir que toda a segurança do sistema ao nível da privacidade, controlo de acesso, acesso granular aos dados, autenticação, etc. seja conseguido.

4.2.6 ENTIDADES CONSUMIDORAS

Como forma de limitação, as entidades que consomem os dados terão de ter algum tipo de relação com os utilizadores que produzem os dados. Desta forma, não podem existir entidades terceiras sem qualquer relação com os utilizadores produtores de dados, a requisitarem acesso aos mesmos. Isto acontece porque existe informação de registo que o utilizador necessita de fornecer à entidade consumidora para que esta saiba como proceder à construção do pedido de acesso aos seus dados.

Assim, na arquitetura que é proposta, o utilizador terá de ser, tipicamente, um cliente da entidade consumidora de dados. Como consequência, esta última terá de ser, tipicamente, um fornecedor de algum tipo de serviço aos seus clientes. Estando esta relação estabelecida onde é possível que o cliente interaja com o seu fornecedor de serviços de forma direta (sem intermédio da plataforma) para efeitos de registo e outras operações, é possível que o utilizador possa controlar o acesso com um nível de granularidade mais alto aos seus próprios dados feito pelo seu fornecedor, evitando que este último se apodere de informação valiosa útil para atacantes, quebrando a privacidade do utilizador.

capítulo 5

Solução Proposta

Neste capítulo é feita a descrição da solução que é proposta de forma a resolver os principais problemas de segurança existentes numa plataforma IoT.
Para além de resolver os problemas de segurança abordados, é pretendido ainda que a arquitetura que é descrita de seguida procure respeitar os requisitos que foram, também, abordados. São aqui apresentadas justificações acerca das várias opções que foram tomadas na elaboração desta solução.

5.1 visão geral da arquitetura

Nesta secção irá proceder-se uma visão geral sobre a arquitetura que é proposta nesta dissertação.

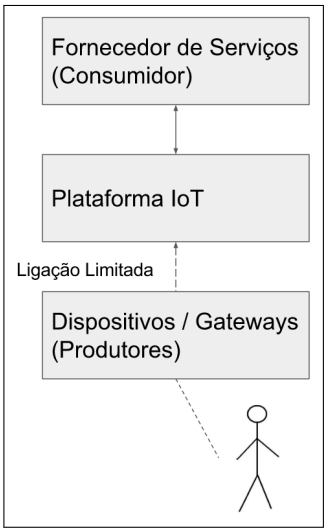

Figura 5.1: Esquema de alto nível de uma arquitetura para uma plataforma IoT convencional

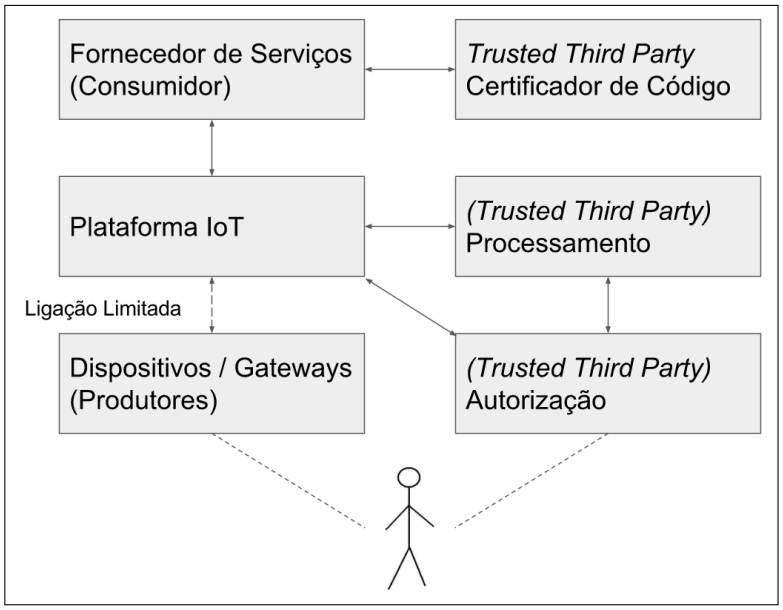

Figura 5.2: Esquema de alto nível de uma arquitetura para uma plataforma IoT com mecanismos de segurança de dados aplicados

Na Figura 5.1 está presente um esquema de uma arquitetura de uma plataforma IoT convencional, isto é, sem mecanismos de segurança de dados aplicados.

Como foi explicado na Secção 2.1, todos os sistemas IoT têm três componentes a muito alto nível: os produtores de dados, uma plataforma IoT e os consumidores de dados. Efetuando a ligação ao esquema apresentado na Secção 2.3, os produtores situam-se na camada mais baixa do sistema (primeira camada do modelo de referência da Cisco), os consumidores nas camadas mais altas do sistema (camada 6 e 7 do modelo de referência) e a plataforma IoT que engloba todas as restantes camadas do modelo de referência.

Na Figura 5.2 está presente um esquema de arquitetura da plataforma IoT onde foram aplicados mecanismos de segurança ao nível dos dados. Todos os componentes presentes na Figura 5.1 necessitaram de ser alterados e foi necessário adicionar 3 entidades/componentes adicionais. De notar que a plataforma IoT representada nestas figuras, inclui vários componentes elementares presentes na Figura 2.4. É o caso da base de dados, domínio de rede, componente de enriquecimento e barramento de mensagens. Nenhum destes componentes necessitou de ser alterado, no entanto, a plataforma IoT sofreu adições de novos componentes.

A essência nesta solução está no facto de não poder dar o acesso total aos dados mesmo que seja aos seus consumidores. Isto porque, se tal acontecer, a privacidade dos utilizadores donos destes dados pode ser posta em causa, pois nem os consumidores dos dados, nem a plataforma IoT podem ser, muitas vezes, confiáveis. Para tal é necessário efetuar uma transformação dos dados originais para que o consumidor não

tenha acesso aos dados na sua íntegra, mas sim a uma determinada transformação desses dados que, mesmo não sendo os originais, pode proporcionar informação útil ao consumidor enquanto que, ao mesmo tempo, o acesso detalhado aos dados é evitado. Este processamento sobre os dados existente de forma a poder fornecer o anonimato desejado poderia ser efetuado de forma automática através de mecanismos de anonimato como k-anonimato. Mesmo que a utilização destes mecanismos possa ser fiável no que toca à manutenção da privacidade, não é garantido que os dados sejam realmente úteis para os consumidores. Para além disso, existe ainda o problema de o utilizador não dar a autorização necessária para que o consumidor tenha acesso aos dados. Finalmente, outro problema do k-anonimato é o facto de este não funcionar sobre dados cifrados. Tal implicaria que os dados chegassem à plataforma em claro, o que se está a tentar evitar nesta dissertação.

A transformação destes dados terá de ser efetuada através de um certo programa (código). Este terá de ser elaborado pelo próprio consumidor de dados uma vez que pode ser malicioso ou pode criar uma transformação/generalização dos dados não suficientemente correta e não respeitando a privacidade do utilizador. Assim, este programa terá de ser validado. Esta validação é feita através de uma certificação desse código, adicionando opcionalmente uma asserção assinada (Secção 5.2) contendo certas condições de processamento/utilização dos dados também certificadas. A inclusão desta asserção não foi implementada, no entanto a sua inclusão é simples de ser efetuada caso a entidade que valida o código necessite de estabelecer certas restrições de utilização do código. Esta certificação de código terá de ser efetuada por uma entidade confiável terceira, pois não poderíamos atribuir este papel a outro componente/entidade. Ao efetuar uma distribuição de poderes como esta, estamos a aumentar o nível de segurança de todo o sistema, sendo que um atacante que tome controlo sobre uma destas entidades não poderá comprometer a privacidade dos donos dos dados que estão a ser atacados.

De forma a executar o código certificado, deve existir uma entidade para o fazer: o componente de processamento ilustrado na Figura 5.2. Esta terá acesso aos dados em claro, uma vez que é a entidade que, exclusivamente, irá fazer o processamento e transformar os dados. Este componente não necessita de ter interface com o utilizador e tem o único propósito de efetuar esse processamento e enviar os resultados de volta. Este componente terá de poder criar ambientes de execução para a execução de código não permitindo o acesso ao sistema fora do seu ambiente de execução. Este pode pertencer e ser registado no domínio seguro do utilizador ou então pertencer a uma entidade terceira confiável onde o utilizador terá que alugar e registar-se pelo serviço a fim de utilizar recursos computacionais para efetuar essa execução. Uma vez que se pretende distribuir os poderes sobre os diversos tipos de informação por entidades/componentes diferentes, caso tais componentes não possam pertencer ao domínio do utilizador, convém que uma

TTP seja utilizada para tal.

O consumidor necessita de autorização por parte do utilizador dos dados para lhes poder aceder e efetuar o processamento referido acima. Para tal, é introduzido, finalmente, a TTP de autorização (Figura 5.2). Este componente pode pertencer ao domínio de segurança do utilizador. Caso contrário, o utilizador necessitará de aderir a um serviço deste género, pertencente a uma entidade terceira confiável. Este componente ou entidade terá de ter uma interface com o utilizador. Este último irá ter o poder de autorizar e verificar a autenticidade dos registos dos seus equipamentos produtores (*gateways*/dispositivos), de autorizar explicitamente, e com a sua própria intervenção, o acesso aos seus dados privados, terá também de armazenar informação do utilizador (caso seja uma entidade externa confiável) e guardar e associar chaves criptográficas usadas na cifra dos dados com identificadores não persistentes dos seus equipamentos produtores. No caso de uso que se tem vindo a considerar dos medidores de eletricidade inteligentes, caso o utilizador permita que o fornecedor de serviços em que estão registados os equipamentos assim como o próprio utilizador, aceda aos seus dados privados de forma a que estes sejam transformados, é emitida uma asserção Security Assertion Markup Language (SAML), cujas condições de utilização são assinadas. Esta asserção constitui uma prova de que, qualquer que seja o lugar onde esta chegue, determinada entidade tenha acesso a determinados dados utilizando um determinado código certificado e utilizando uma determinada TTP de processamento para o efeito. Esta asserção será mais tarde validada pelo componente de processamento.

A arquitetura que aqui é proposta introduz vários componentes TTPs. No entanto, existem dois componentes que não necessitam de ser uma entidade terceira por si só. Estes dois componentes são o de processamento e o de autorização. Como será explicado com detalhe, mais à frente, caso estes dois componentes estejam sob o controlo dos utilizadores fisicamente e, portanto, dentro de um perímetro seguro, o sistema ganha uma maior segurança e não é necessário confiar em entidades externas adicionais.

O componente TTP certificador de código terá de ser uma entidade terceira confiável pelo utilizador e pela TTP de processamento que o utilizador tem sob controlo.

5.2 SAML

O SAML [50] foi um protocolo escolhido para efeitos de autorização. Uma asserção consiste na emissão de um documento no formato XML que contém informações sobre a origem, o destinatário, instantes de tempo e outras condições opcionais que é possível adicional. Todo este documento é assinado o que permite que os elementos que necessitam de verificar esta autorização, a verifiquem de forma fácil e segura.

Existiam, tipicamente, duas opções principais para efetuar autorização neste tipo de arquiteturas seguras. Uma delas era utilizando SAML e outra era utilizando o OAuth [31].

O OAuth é um protocolo baseado em *tokens* onde um Identity Provider (IdP) emite um código caso o utilizador forneça autorização de acesso à sua informação do IdP num fornecedor de serviços. Este *token* é utilizado por um cliente quando comunica diretamente com um fornecedor de serviços, que terá de comunicar diretamente com o IdP de forma a poder validar o *token*. Assim, o OAuth deve ser aplicado em situações onde se delega uma determinada entidade a atuar por sua conta.

O SAML é um protocolo baseado em asserções onde um IdP emite uma asserção assinada (podendo conter certas condições de utilização) caso o utilizador forneça autorização de acesso à sua informação do IdP em um fornecedor de serviços. Esta asserção é utilizada por um cliente quando comunica diretamente com um fornecedor de serviços. Este último não precisará de comunicar novamente com o IdP de forma a validar a asserção (no entanto uma comunicação prévia deverá ter sido feita de forma a obter uma chave pública). A asserção pode ser validada simplesmente verificando a assinatura do XML emitido e verificando os termos e condições presentes na asserção em causa. Uma vez que são utilizadas assinaturas, a autenticação é garantida, para além da autorização e da inclusão de restrições. No fundo, esta asserção pode atuar como um *token*. A diferença está no facto de que a asserção é assinada promovendo a autenticação e permite inclusões de condições e restrições. Assim, o SAML deve ser aplicado em situações onde se pretende autorizar que uma determinada entidade tenha acesso a determinada informação de acordo com certas restrições.

Tendo em conta estas informações, foi decidida a utilização de asserções SAML como mecanismo de autorização nesta arquitetura.

5.3 COMUNICAÇÃO ENTRE COMPONENTES

Os vários dispositivos e componentes necessitam de ser autenticados entre si, nomeadamente aqueles cujas comunicações são cruciais para a garantia de segurança que se pretende obter. Este facto é particularmente necessário em plataformas onde existe uma grande quantidade de comunicação entre componentes e dispositivos, de modo distribuído ou não, como acontece no caso da IoT. Tal é necessário para evitar ataques *man-in-the-middle* onde um invasor se insere no meio das comunicações entre dois dispositivos/componentes.

Para além deste tipo de ataques, a existência de autenticação também permite que uma aplicação impostora se tente fazer passar ou tome o papel de uma outra aplicação

legítima existente. A nível de segurança, todos estes aspetos são importantes, tornando a autenticação num mecanismo de segurança obrigatório. Embora a autenticação de produtores, de consumidores e dos componentes incluídos na plataforma IoT não seja um objetivo principal (exceptuando a autenticação que existe ao nível de comunicação entre o *gateway* e a TTP de autorização, explicada na Secção 5.5) e já saia fora do foco central desta dissertação, este é um requisito de segurança importante em todas as plataformas seguras. A solução que foi encontrada para lidar com este problema na arquitetura proposta é a utilização de protocolos Secure Sockets Layer (SSL)/TLS em todas as comunicações entre componentes/dispositivos onde é possível garantir que apenas os componentes fidedignos podem estabelecer ligações entre si.

Os comandos, dados e meta-dados convém estarem também protegidos contra invasores para que a menor quantidade de informação interna possível não seja liberada para o exterior, garantindo a confidencialidade dos dados. Tal é também conseguido com a aplicação dos protocolos SSL/TLS nos canais de comunicação.

Por fim, a integridade dos dados é também importante (os protocolos SSL/TLS também a podem garantir). A integridade dos dados nas comunicações, caso seja garantida, torna todas as comunicações fiáveis, sendo que a fiabilidade de uma plataforma IoT é também um requisito importante a ser conseguido. A falta de fiabilidade dos dados resulta numa má aceitação daquilo que são os padrões de qualquer sistema (não só IoT). A fiabilidade é ainda mais essencial no contexto do IoT. Tomando como exemplo o caso de uso principal desta dissertação (medidores de energia elétrica inteligentes), imaginemos que a fiabilidade dos dados não se mantinha, ou seja, estes podiam ser alterados por terceiros (por exemplo através de um ataque *man-in-the-middle*) no ato de transferência de dados pelo canal de comunicação atacado. Caso exista algum atacante interessado em prejudicar os valores de consumo de eletricidade de uma determinada residência, este poderia proceder, através deste tipo de ataque, a uma alteração dos valores de consumo de energia que vão sendo comunicados ao *gateway* (embora este esteja numa zona segura) ou à própria rede IoT. Tomando como exemplo o caso de uso da domótica, caso um utilizador utilize a sua aplicação de controlo remoto para ligar o aquecimento de um quarto a uma determinada temperatura, este valor tem de chegar ao equipamento alvo na sua íntegra, sem que seja alterado para uma temperatura diferente através de um qualquer ataque efetuado ao meio de comunicação.

5.4 COMPONENTES DA SOLUÇÃO

Na Figura 5.3 está presente um diagrama com os componentes da arquitetura que se propõe para uma plataforma IoT segura que preserva a privacidade dos dados dos

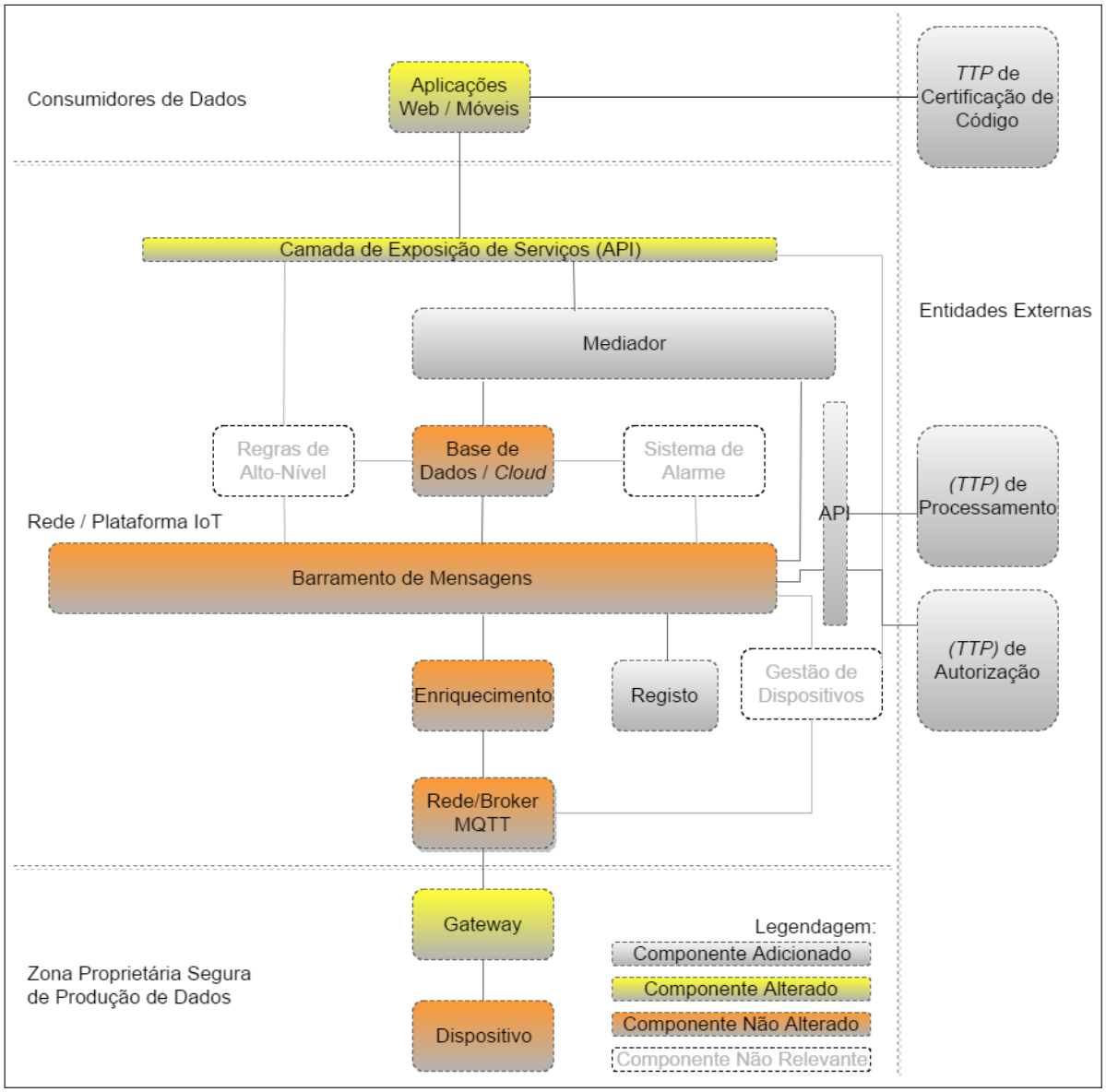

Figura 5.3: Componentes da arquitetura para uma plataforma IoT segura que se propõe

utilizadores, fornecendo a possibilidade aos últimos de poderem controlar o acesso aos seus próprios dados (algo que era impossível numa plataforma IoT convencional e não segura) e a possibilidade de não fornecer os dados produzidos, na sua íntegra, aos consumidores. Para posteriormente implementar os mecanismos de segurança que resolvem os problemas de segurança identificados, foi utilizada uma arquitetura IoT já desenvolvida (Figura 2.4) onde foram adicionados outros componentes e modificando os existentes. Relativamente ao componente pertencente ao domínio de rede, o mesmo não sofreu quaisquer alterações pelo que será considerado como pertencente à plataforma IoT, daqui em diante.

Como se pode observar, os diversos componentes encontram-se divididos pelas três camadas de alto nível que foram antes referidas: produtores, plataforma IoT e

consumidores. Os componentes representados a branco são componentes presentes numa plataforma IoT convencional que não têm qualquer tipo de interação ou ligação com os mecanismos que se propõe adicionar. Os elementos com fundo laranja são elementos que não sofreram alterações. Os elementos com fundo amarelo são elementos que sofreram alterações e os restantes são novos elementos/componentes/entidades adicionadas para que a segurança no sistema IoT se concretize.

Na parte inferior da imagem observa-se os Dispositivos e *Gateways*. Estes dois componentes necessitam de estar localizados numa zona segura. Para que tal seja possível, quer estes dois componentes, quer a ligação que é feita entre eles deverá estar num local seguro onde seja impossibilitado o acesso físico por parte de terceiros a estes componentes e meio utilizado na comunicação. No contexto do caso de uso de uma plataforma IoT que está a ser utilizado nesta dissertação (medidores de consumo de energia elétrica inteligentes), uma zona fisicamente protegida mais óbvia é a própria residência cujo consumo de energia elétrica será monitorizado. Neste caso, o Dispositivo seria o próprio medidor/contador inteligente. A necessidade destes dois componentes e a ligação entre eles terem que estar presentes numa zona segura passa pelo facto de que os dados que passam nessa ligação com origem o dispositivo produtor e com destino o *gateway*, não estão protegidos e, por isso, passam em claro. Isto pode verificar-se caso os dispositivos/sensores que tratam de colher os dados de consumo sejam tão simples que não possuam capacidades de criação de ligações SSL/TLS (ou utilizando outros protocolos de comunicação) de forma a assegurar o envio de dados cifrados através de um canal de comunicação seguro. De notar que a existência dos *gateways* pode ser opcional. Neste caso, os dispositivos deverão ter que ser alterados e possuir as mesmas capacidades de segurança dos *gateways* que foram, na arquitetura proposta, introduzidas.

Na parte superior da imagem estão representados os consumidores dos dados. Os consumidores são, normalmente, aplicações Web ou móveis normais. Existe uma limitação para estes consumidores: estes têm que estar relacionados com os dispositivos/*gateways* dos utilizadores, com os próprios utilizadores e com a entidade externa de autorização. O estabelecimento dessa relação necessita de acontecer através de comunicações de registo. Só assim é que será possível dar controlo ao utilizador dos consumidores dos seus dados, permitindo-lhe autorizar ou não o acesso à sua produção. De notar que o acesso à produção de dados é um requisito de segurança. Tal não era possível utilizando uma plataforma IoT convencional, onde o utilizador não tinha qualquer controlo sobre a sua produção.

Entre os consumidores e os produtores de dados encontra-se a plataforma IoT. Este conjunto de componentes tem como principal missão: gerir os diferentes dispositivos e entidades que a usam para comunicar, fornecer serviços de armazenamento dos dados,

transformá-los, aplicar regras, tratar de toda a lógica de comunicações entre os diversos componentes/dispositivos.

Finalmente, na Figura 5.3, está representada uma área constituída por componentes externos. Estes componentes pertencem, opcionalmente, a uma entidade terceira (à exceção da TTP de certificação de código que tem de obrigatoriamente ser uma TTP). No entanto, irá ser tratado o componente de processamento e o de autenticação como se fossem TTPs de forma a analisar o pior caso, o caso onde é necessário estabelecer uma relação de confiança com entidades externas. A TTP de processamento irá ter acesso aos dados em claro, no entanto, é assumido que a informação inerente não possa ser associada a um determinado utilizador, uma vez que a informação de utilizadores nunca é comunicada a esta entidade. No entanto, é conveniente que esta entidade seja de confiança de forma a evitar que os dados sejam manipulados de forma errada e não utilizando o código certificado mencionado.

Os componentes externos à plataforma podem não estar disponíveis para responder prontamente em determinado momento. Ou porque se encontram sobrecarregados com processamento ou porque pode não ser possível retornar um resposta prontamente e em tempo útil. Este último pode ocorrer, por exemplo, quando é efetuado um pedido de processamento dos dados à TTP de processamento. Caso a quantidade de dados seja enorme, o processamento deverá não ser tão rápido quanto isso o que faria com que o iniciador deste pedido necessitasse de esperar em bloqueio pela resposta. Devido a estes factos e como solução, estes pedidos REST (APIs utilizadas na arquitetura proposta) que são feitos tendo a plataforma IoT como interveniente, são efetuados de forma assíncrona. Isto é, a entidade que recebe o pedido deve enviar uma resposta REST logo que seja possível e, só após essa resposta, é que procede à execução das tarefas necessárias inerentes ao pedido. Quando os resultados estiverem prontos, esse componente inicia uma nova comunicação de forma a enviar a resposta tão aguardada pelo componente que fez o pedido original.

Sempre que se refere a um identificador, este é geralmente um UUID, ou seja, um identificador universal único, a não ser que seja referido explicitamente que não o é.

Nas secções seguintes serão descritos os papéis dos componentes relevantes para a segurança do sistema, ou seja, os componentes que foram adicionados e os componentes que foram modificados.

A visão sobre a arquitetura pode ficar mais clara na Secção 5.5 quando forem apresentadas diferentes operações assim como o fluxo dos dados e o processamento sobre eles efetuadas pelos diversos componentes/entidades.

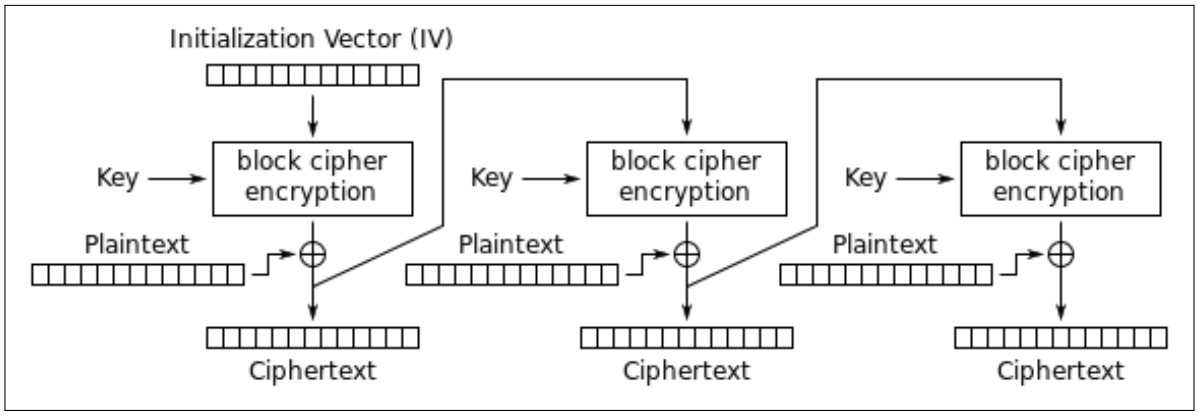

Figura 5.4: Diagrama do funcionamento do modo de cifra CFB na operação de cifra, retirado de Wikipedia[1]

5.4.1 DISPOSITIVO *gateway*

Este componente necessita de ser alterado de forma a que os mecanismos de segurança que se propõe sejam implementados em plataformas IoT.

O *gateway* tem um propósito essencial no que toca à arquitetura em questão. Ele deve aplicar métodos criptográficos sobre os dados que recebe, em claro, dos dispositivos produtores. O método criptográfico aplicado é uma cifra simétrica utilizando o algoritmo AES em modo Cipher FeedBack (CFB), com uma chave simétrica de 256 bits. Antes da operação de cifra dos dados provenientes dos dispositivos produtores, é criado um *digest* dos mesmos. A este *digest* são concatenados os dados originais em claro e de seguida é que é efetuada a operação de cifra. O algoritmo de cifra AES foi escolhido uma vez que, no momento de execução desta dissertação, este é um dos algoritmos onde ainda não foram encontradas vulnerabilidades e, portanto pode-se afirmar como seguro.

O modo de cifra CFB foi escolhido, principalmente, por causa do facto de ser uma cifra por blocos onde o próximo bloco de cifra a ser cifrado depende do resultado da cifra do bloco anterior (Figura 5.4). Desta forma, a decifra é efetuada executando a operação de decifra do algoritmo AES sobre cada bloco de texto cifrado. Tal fornece a possibilidade da decifra ser efetuada paralelamente (maior rapidez) e de apenas decifrar determinados blocos desejados nos casos em que não se pretende obter toda a informação (maior flexibilidade). Poder-se-ia por em causa a aplicação deste algoritmo neste tipo de dados, uma vez que os formatos dos diferentes dados recebidos no *gateway* são muito distintos. No entanto, para além de ser utilizado um Initialization Vector (IV) diferente a cada momento de cifra, é concatenado aos dados, o *digest* referido. No processo de cifra, o *digest* é o primeiro a ser cifrado. Dada à aleatoriedade do mesmo, os blocos de texto cifrado correspondentes ao *digest* serão também aleatórios. Esta aleatoriedade propaga-se ao longo do algoritmo tendo como consequência a aleatoriedade dos blocos de

[1] `https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation`

texto cifrado correspondentes aos dados. Desta forma, é possível evitar certos ataques como análise e comparação do texto cifrado com textos cifrados e/ou textos em claro já conhecidos. A chave simétrica utilizada na cifra não é gerada no próprio *gateway* e tem de ser pedida ao TTP de autorização associado. Esta associação é feita através de uma operação de registo efetuada pelo utilizador, no serviço que pretende aceder aos seus dados. Este transporte de chave é efetuado de modo seguro, cifrada com uma chave pública do *gateway*. Esta chave é transmitida de forma segura e com autenticação à TTP de autorização como será detalhado mais tarde neste documento.

De momento, estes foram os algoritmos e modo de cifras escolhidos. No entanto, a arquitetura e o processo de decifra destes dados foi pensado de forma a que, em trabalho futuro, se possa implementar um mecanismo de escolha de algoritmos. Essa escolha poderia ser feita pelo próprio utilizador (caso os seus componentes (*gateways* ou dispositivos) tivessem as bibliotecas necessárias para tal) no processo de associação e registo do *hardware* do utilizador para com o serviço consumidor dos dados, respetivo. Esta operação estará detalhada na Secção 5.5. Como é o código certificado a ser executado na TTP de processamento que irá tratar de executar a decifra, a informação sobre os algoritmos utilizados pode ser obtida juntamente com as chaves da TTP de autorização associada e pertencente ao utilizador.

Foi tomada a opção de executar o processo de cifra dos dados, fazer uso de um TTP de processamento para executar o código que irá processar os dados em claro e fazer uso de um TTP de autenticação separado, uma vez que este dispositivo, embora, hoje em dia já seja mais potente e tenha maiores capacidades e condições para efetuar processamento sobre dados, não tem capacidades suficientes para receber uma enorme quantidade de dados da plataforma e com periodicidade alta, pois encontra-se muito limitado no que toca ao método de comunicação. Como já foi referido, tipicamente, as *gateways* fazem uso de dados móveis recorrendo a um cartão Subscriber Identification Module (SIM), devido à incerteza geográfica de colocação da generalidade destes componentes. Para além disso, os operadores de rede podem restringir o número e largura de banda de comunicações tendo como origem a rede móvel e destino os dispositivos móveis. Como consequência, os dados e o código de anonimato certificado não poderiam ser comunicados a este dispositivo para que decifre os dados e lhes aplique funções de generalização.

Uma vez que não é este componente que decifra os dados, a chave será gerada pela TTP de autorização (Figura 5.5). Assim, o *gateway* irá necessitar de efetuar um pedido a esta TTP para que este gere e retorne uma chave simétrica. O *gateway* não possui nenhuma informação acerca das TTPs, uma vez que na grande maioria das vezes não possuem uma interface com o utilizador para que o mesmo os possa configurar de forma livre. É a plataforma IoT que ganha conhecimento desta informação no momento de

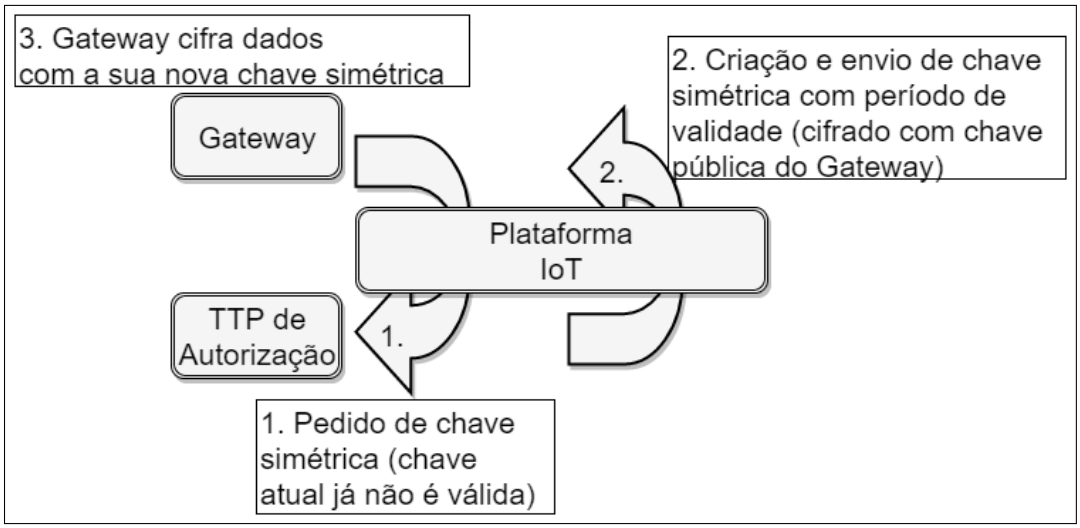

Figura 5.5: Diagrama representativo do pedido de chaves simétricas efetuado pelo gateway *à TTP de autorização associada*

registo. O *gateway* apenas limita-se a pedir uma chave simétrica quando necessita ou quando uma operação para tal é despoletada sobre ele a partir de uma TTP de autorização associada. Os detalhes deste protocolo serão referidos na Secção 5.5.

No início, depois de ser efetuado o registo, o *gateway* necessita de pedir uma chave simétrica a ser utilizada nos dados que vai recebendo. Essa chave é válida apenas durante um determinado período de tempo configurável pelo utilizador na TTP de autorização.

Na comunicação da chave, terá lugar um mecanismo de cifra assimétrica de forma a fornecer proteção adicional. Juntamente com esta chave simétrica, a TTP envia, na resposta ao pedido da chave, um período de tempo em que a chave é válida e pode ser utilizada para cifra de dados. A cifra de dados utilizando sempre a mesma chave de cifra é perigosa, sobretudo quando esta abandona o dispositivo em que foi gerada e também devido ao facto da TTP poder estar localizada remotamente e o seu respetivo *hardware* poder ser acedido por estranhos. Para além disso pode dar pistas a um atacante sobre a chave, caso este obtenha acesso aos dados cifrados aquando da sua transmissão. O atacante, poderá então comparar vários textos cifrados e conseguir encontrar, através de criptanálise, a chave utilizada ou parte dela. A probabilidade de sucesso deste tipo de ataque é um pouco diminuído com o facto da utilização de diferentes IVs no método de cifra simétrica aplicado. O *gateway* ao receber todos estes dados da TTP, procede à decifra da chave simétrica utilizando a sua chave assimétrica privada, previamente gerada, cifra os dados com a chave simétrica e envia o texto cifrado com destino a plataforma IoT para que os dados sejam armazenados.

O *gateway* não precisa de estar constantemente a efetuar o pedido de chaves simétricas à TTP de autorização. Desde que o instante de tempo atual ainda esteja dentro do

intervalo de tempo em que a chave simétrica é válida, o *gateway* pode proceder à cifra dos dados utilizando essa chave e enviar, posteriormente, o texto cifrado para a plataforma IoT. Caso o prazo de validade da chave tenha esgotado, é feito um pedido de chaves à TTP de autorização. Se o *gateway* optar por continuar a utilizar uma chave de cifra não válida, os dados não poderão ser acedidos no futuro e os dados produzidos serão perdidos.

O *gateway* necessita de se registar e de assinalar a presença de um novo dispositivo na plataforma IoT assim que se liga. Desta forma, na inicialização, o *gateway* irá ter de enviar uma mensagem de registo para a TTP de autorização associada previamente (detalhes referidos na Secção 5.5). Nesta mensagem terá de comunicar a sua chave pública (esta será criada sempre que o dispositivo inicia pelo que não é persistente). Para evitar que nenhuma outra entidade (nomeadamente a plataforma) tenha alterado esta chave na sua passagem (por exemplo, através de um ataque *man-in-the-middle*), é utilizado um mecanismo que dificulta essa mesma alteração da chave pública cuja chave privada respetiva seja válida e permita ao atacante decifrar as chaves simétricas que são, posteriormente, cifradas com a chave pública em questão. Este mecanismo consiste no envio por parte do *gateway* de um conjunto aleatório de operações efetuadas sobre a chave pública legítima produzindo um código relativamente pequeno (4 caracteres). Ao efetuar as mesmas operações do lado do TTP produzirá o mesmo código, caso não tenha existido interferência, validando a chave pública comunicada. Detalhes serão referidos na Secção 5.5. De forma a mostrar este código, os *gateways* deverão possuir uma interface muito simplificada para com o utilizador. Tal pode ser feito através de um ecrã OLED, por exemplo. Assim o utilizador poderá comparar os códigos mostrados pelo *gateway* e pela TTP de autorização. Caso sejam iguais, o utilizador poderá aceitar o registo do seu próprio dispositivo no momento de inicialização. De notar que, quando o *gateway* é reinicializado, este efetuará um novo registo, utilizando opcionalmente um novo identificador do mesmo. Neste caso o utilizador terá de ter em sua posse essa informação previamente para que possa associar o *gateway* com a TTP de autorização, previamente. Processo este que será explicado mais adiante. No entanto, esta troca de identificadores terá de ser gerada pelo *gateway* e comunicada ao utilizador (por exemplo através do ecrã LED mencionado) previamente para que este mecanismo funcione. A troca ou rotação de identificadores dos *gateways* ou dispositivos produtores poderá, eventualmente, fazer parte de trabalho futuro. De notar que, caso o *gateway* se registe novamente com o mesmo identificador (devido a, por exemplo, ter sido reiniciado), a TTP de autorização deverá verificar que tal aconteceu e não requerer que o utilizador interaja com a TTP de forma a aceitar o pedido de registo, sendo que esta aceitação deverá ser automática. Configurações relativamente a este aspeto podem existir na própria TTP de autorização. Desta forma, está-se a dar o encargo de autenticação entre

o *gateway* e a TTP de autorização ao próprio utilizador. Apenas este poderá autorizar a autenticação verificando e comparando dois simples códigos.

5.4.2 MEDIADOR

O Mediador é um componente novo introduzido nesta arquitetura comparativamente a uma arquitetura para uma plataforma IoT convencional e não segura. Este componente não possui papéis muito complexos, tratando-se, fundamentalmente, de um *proxy*. Sempre que uma aplicação consumidora pretenda aceder aos dados armazenados pela plataforma, ao invés de ser feito um acesso direto ao componente de armazenamento com o intuito de recuperar dados, os pedidos passam primeiro por este componente chamado de Mediador. Ao receber um pedido de consulta de dados, este componente recorre à TTP de autorização de forma a obter a autorização expressa do utilizador através da receção de uma asserção SAML (Secção 5.2). Caso a aplicação em causa não possua a autorização necessária para aceder a estes dados devido ao pedido de autorização efetuado pelo Mediador à TTP, o acesso aos dados termina aqui. Para além da consulta efetuada à TTP para efeitos de autorização. Para que a apresentação dos dados seja feita sem que a privacidade dos donos dos dados seja comprometida, é implementado nesta arquitetura um mecanismo de generalização, transformação e anonimato de dados conseguido através da execução de código certificado. A execução é feita pelas TTPs de processamento. No entanto, o código compilado e meta-dados são armazenados no Mediador mediante um registo de código compilado prévio entre a aplicação consumidora e o Mediador (pertencente à plataforma). Após o Mediador armazenar o código compilado e os respetivos meta-dados, este retorna um identificador para o consumidor. Os meta-dados referidos incluem informação sobre chaves públicas, assinaturas e ouras informações úteis relacionadas com o código certificado a ser executado e aprovados pela TTP de certificação de código. Estes são incluídos num manifesto presente num ficheiro.

No momento de acesso aos dados (Figura 5.6), é imperativo que seja indicado, através de identificadores do código gerados pelo Mediador no momento de registo, para a execução do código certificado desejado, juntamente com informação dos *gateways*/dispositivos de onde se pretende obter os dados, um intervalo de tempo que restringe o número de dados produzidos desejados e ainda informação sobre as TTPs que irão estar envolvidos neste pedido.

O Mediador procede ao requisito de autorização à TTPs de autorização incluída. Caso tal acesso seja concedido por cada TTP, o Mediador procede à recuperação dos dados cifrados em questão comunicando diretamente com o componente de armaze-

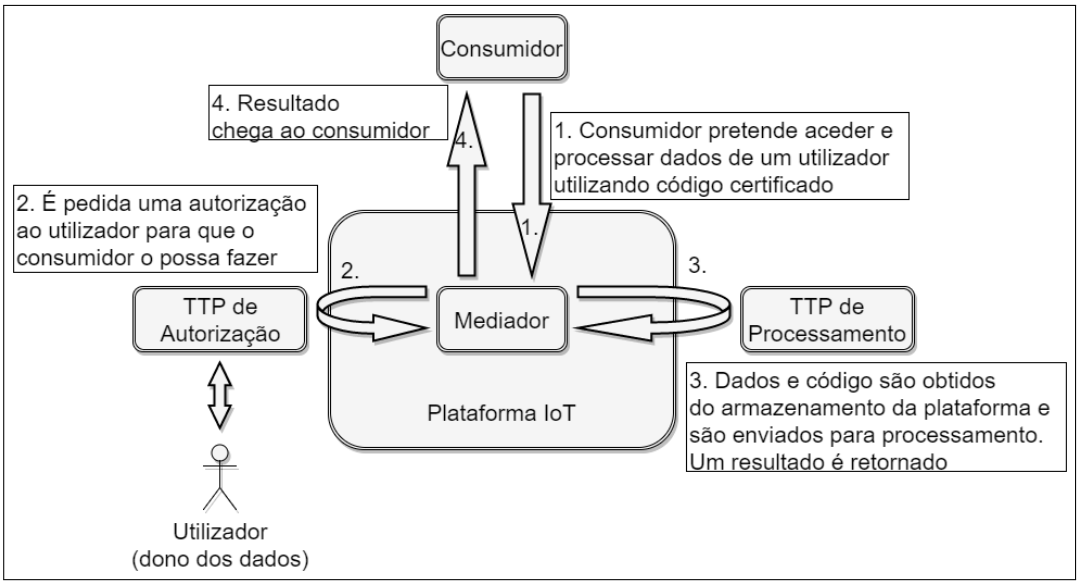

Figura 5.6: Diagrama representativo do processo, a alto nível, do acesso aos dados por parte do consumidor e da obtenção de uma transformação dos dados originais do utilizador dono dos dados

namento de dados da plataforma IoT. Estes dados cifrados retornam ao Mediador temporariamente, sendo que este irá, agora, enviar quer os dados cifrados quer o código compilado certificado para ser executado na TTP de processamento, cujo identificador foi mencionado no pedido, A TTP de processamento após efetuar o seu processamento e execução deste código, retorna com uma resposta contendo dados generalizados cifrados. Esta cifra é efetuada pela TTP de processamento e é uma cifra híbrida. Isto é, os resultados do processamento são cifrados simetricamente com uma chave simétrica gerada no momento. Esta chave simétrica é depois cifrada assimetricamente com a chave pública do destinatário (o consumidor). A chave pública do consumidor está incluída nos meta-dados e é acessível pela TTP de processamento.

O Mediador, ao receber a resposta da TTP, procede ao envio das mesmas para a aplicação consumidora. Esta irá decifrar os dados e irá apenas ter acesso à transformação e generalização aplicada e não aos dados originais assim como foram produzidos e armazenados, fornecendo uma vista ou abstração sobre os dados, recorrendo a mecanismos de generalização personalizada e anonimato garantindo um nível alto de privacidade.

Foi considerado um caso de uso adicional onde é feito um processamento adicional sobre todos os resultados independentemente processados pela TTP de processamento. O consumidor pode requerer dados de diversos dispositivos produtores/*gateways* que estejam associados a diferentes TTPs de autorização. De forma a obter toda esta informação, o consumidor poderia simplesmente efetuar os vários pedidos de forma independente. No entanto, como forma de aumentar e preservar ainda mais a segurança dos dados e privacidade do utilizador, os resultados obtidos independentemente, neste

caso de uso, não seriam devolvidos ao consumidor. Ao invés disso, estes resultados seriam enviados para um TTP de processamento agregador adicional. Este processamento agregado seria feito sobre os resultados individuais dos dados dos diversos dispositivos produtores. Este resultado final seria retornado para o Mediador e, só então, é que o Mediador iria comunicar com o consumidor. Uma vez que existem duas camadas de processamento diferentes, seria necessário que existissem dois tipos de código a serem registados - um para processamento individual por cada TTP de processamento e outro para processamento agregado. Tal foi, a determinado momento considerado, no entanto não foi implementado e foi adicionado a possível trabalho futuro e como um caso de uso possível da arquitetura aqui proposta.

5.4.3 TTP DE AUTORIZAÇÃO

Este componente é um novo componente introduzido na arquitetura. Os seus principais objetivos são a geração de chaves simétricas para que os dados possam ser cifrados e armazenados pela plataforma IoT, a sua gestão, o tratamento e controlo de acesso e o controlo e gestão dos dados dos utilizadores. Este componente pode estar num telemóvel sempre presente com o dono dos dados, num computador na residência do mesmo ou numa máquina virtual remota contratada a uma entidade externa confiável (TTP). Podem existir vários TTPs de autorização associados aos diversos *gateways*.

Caso este componente seja uma máquina localizada numa rede privada atrás de uma Network Address Translation (NAT) [65], não pode ser a plataforma a iniciar as comunicações com os componentes de autorização e processamento, pois a rede NAT não permite que comunicações sejam iniciadas externamente. Ao invés disso, terão de ser estes componentes, através de mecanismos de *polling*, a verificar se a plataforma tem pedidos pendentes a serem processados. No caso da comunicação direta entre os componentes de processamento e autorização (i.e. no pedido de chaves simétricas), a mesma deverá ser feita dentro da respetiva rede privada atrás de NAT. Se o equipamento for um telemóvel, tais comunicações poderão ser efetuadas através de notificações *push* [41].

Uma vez que, muitas vezes, os componentes de autorização e processamento podem não estar acessíveis, ou porque se encontram desligados ou sobrecarregados com muitos pedidos e processamento, a plataforma tem de implementar mecanismos de deteção de tais situações, armazenar os pedidos pendentes e ir tentando comunicar com os mesmos através de *polling*. Uma vez que se está a considerar o pior caso onde estes componentes pertencem a serviços externos confiáveis (TTPs), é assumido que estes componentes foram instalados e implementados com os requisitos de disponibilidade e desempenho

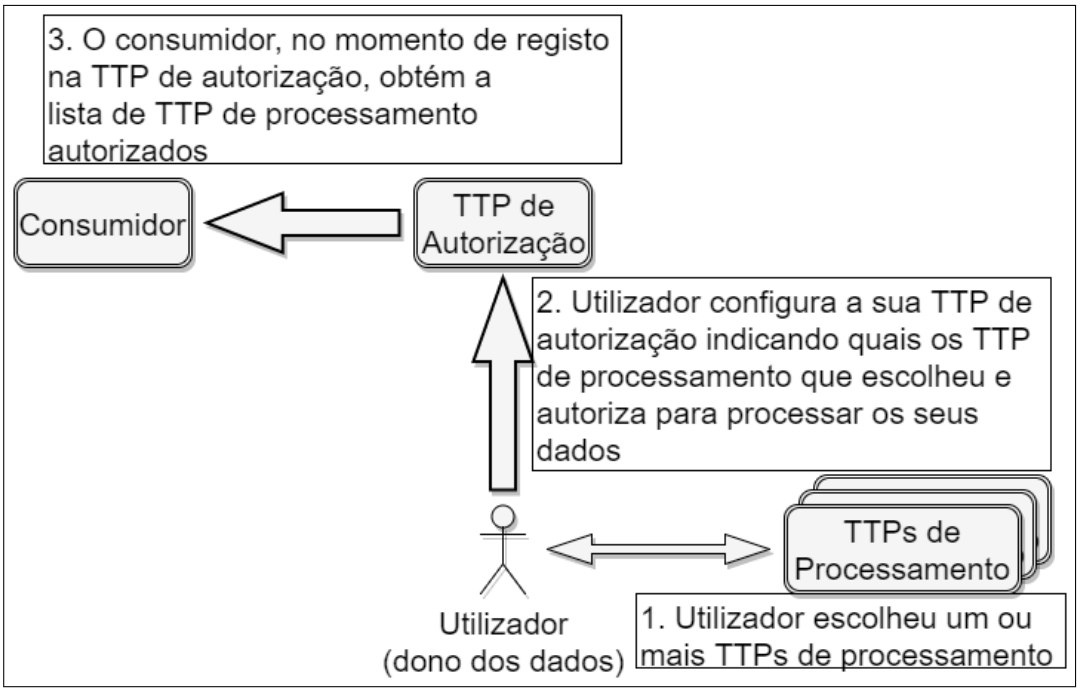

Figura 5.7: Diagrama representativo do processo de configuração da TTP de autorização com a lista de TTPs de processamento permitidos para processamento de dados

em mente. Assim, e para diminuir a quantidade de informação e tráfego de rede que chega à plataforma no caso de existir um grande número de TTPs, foi tomada a decisão da iniciativa de comunicação partir da própria plataforma IoT.

Este componente necessita de ser registado e associado a um determinado utilizador e *gateway* como será possível observar na Secção 5.5. A certa altura neste registo e como possível observar na Figura 5.7, a TTP de autorização irá receber uma comunicação onde é incluído um pedido de registo. Este componente, na resposta, irá devolver uma lista dos TTPs de processamento indicados como autorizados para processamento dos dados em claro. Assim, o consumidor dos dados tem a informação sobre os TTPs de processamento que pode usar quando necessitar de obter dados de um determinado cliente. Esta lista terá de ser configurada pelo utilizador na própria TTP de autorização, antes de ser efetuado qualquer registo desta TTP no consumidor. Desta forma, é dada a opção ao utilizador de escolher quais os serviços ou máquinas, cujos recursos de processamento irão ser utilizados, confiáveis pelo próprio utilizador para aceder aos seus dados em claro. No momento do acesso aos dados, o utilizador terá de autorizar ou não explicitamente e presencialmente o acesso aos dados utilizando um determinada TTP de processamento mencionado no pedido, pelo que a lista devolvida apenas serve como registo dos TTPs preferencialmente disponíveis para o consumidor utilizar. Opcionalmente, a TTP de autorização poderá encarar esta lista de TTPs de processamento como sendo os únicos permitidos. Assim, se for efetuado um pedido

onde é utilizada uma TTP de processamento não permitida, a TTP de autorização pode recusar este pedido automaticamente sem recorrer ao utilizador.

Sempre que o prazo de validade de uma chave simétrica gerada por este componente expira, o *gateway* irá efetuar um pedido a este componente, requisitando uma nova chave simétrica para a cifra dos dados. A TTP de autenticação irá, então, proceder à geração de uma chave simétrica. Juntamente com esta chave, será gerado um intervalo temporal. O utilizador pode configurar a geração deste intervalo manualmente. Ele irá definir o prazo de validade da próxima chave simétrica, ou seja, no momento de cifra de dados por parte do *gateway*, este só pode utilizar a chave gerada, caso o instante de tempo naquele momento pertencer ao intervalo de tempo agora gerado pela TTP de autenticação que corresponde ao prazo de validade da chave. A chave simétrica gerada é cifrada com a chave pública do *gateway* em questão. Desta forma, é adicionada proteção adicional à chave gerada no momento da comunicação da mesma com componentes localizados remotamente onde possa, eventualmente, existir uma rede pública insegura intermediária.

Este componente terá de ter obrigatoriamente uma interface para com o utilizador onde estão presentes ações como: forçar com que o *gateway* registado neste componente necessite de uma chave novamente gerada para que possa continuar a efetuar a cifra dos dados recebidos, efetuar a autorização de registo de um determinado *gateway* verificando o código de prova de chave pública na interface e comparando-a com aquela que é apresentada no *gateway*, efetuar a autorização de acesso aos seus dados por parte do consumidor e efetuar operações sobre as chaves simétricas que estão armazenadas na TTP, como por exemplo, efetuar a transferência de todas as chaves para um dispositivo amovível, eliminar todas as chaves presentes (resultando na perda dos dados, uma vez que não é possível obter os dados sem a respetiva chave de cifra), entre muitas outras possíveis operações.

O TTP de autorização irá ainda enviar para a TTP de processamento em questão e autorizado anteriormente para o processamento dos dados, as chaves secretas que lhe foram pedidas diretamente. A TTP de autorização pode verificar que foi expedida uma autorização ao validar a asserção SAML que ela própria gerou. Esta asserção está assinada e é possível ser verificada no momento em que a TTP de processamento efetua o mencionado pedido de chaves, de forma a que seja possível que ele próprio, utilizando código certificado, as utilize para decifrar os dados e efetuar o processamento de generalização/anonimato sobre eles.

De notar que, embora o utilizador tenha dado autorização para que um dado consumidor possa aceder aos dados de determinados dispositivos, num determinado intervalo de tempo utilizando uma determinada TTP de processamento e executando um determinado código de processamento sobre os dados, o utilizador nunca poderá ter

a certeza do que o consumidor irá fazer com esses dados. Embora sejam, alegadamente, corretos na medida em que não põem em causa a privacidade do utilizador, não existe qualquer forma, nesta arquitetura, de restringir efetivamente como esses dados serão usados por essa entidade e essa questão não será tratada nesta dissertação. No entanto, como já foi mencionado, a TTP de certificação de código poderá, em trabalho futuro, emitir, juntamente com o certificado, uma asserção SAML. Esta poderá seguir até à TTP de autorização para validação de um conjunto de termos e condições (contendo por exemplo uma ligação para uma página de termos e condições de serviço) que a entidade consumidora aceita cumprir. Desta forma, e recorrendo a mecanismos legais poderá vir a ser possível restringir a utilização dos dados processados dos utilizadores por parte da entidade consumidora (Secção 5.4.5).

Caso este componente seja fornecido por um serviço externo, terá de ser implementado um registo de utilizadores banal na TTP.

5.4.4 TTP DE PROCESSAMENTO

Este componente é um novo componente introduzido na arquitetura. Os seus principais objetivos são a execução do processo de decifra dos dados, a execução do código compilado certificado sobres os dados em claro e o pedido das chaves simétricas correspondentes a estes dados. Este componente deve ser uma máquina capaz de efetuar processamento de dados pesado. Assim pode ser, por exemplo, um computador na residência do utilizador ou até um conjunto de recursos (processador/memória) contratados a uma entidade externa confiável.

Caso este componente seja uma máquina localizada numa rede privada atrás de uma NAT, não pode ser a plataforma a iniciar as comunicações com os componentes de autorização e processamento, pois a rede NAT não permite que comunicações sejam iniciadas externamente. Ao invés disso, terão de ser estes componentes, através de mecanismos de *polling*, a verificar se a plataforma tem pedidos pendentes a serem processados. No caso da comunicação direta entre os componentes de processamento e autorização (i.e. no pedido de chaves simétricas), a mesma deverá ser feita dentro da respetiva rede privada atrás de NAT.

Uma vez que, muitas vezes, os componentes de autorização e processamento podem não estar acessíveis, ou porque se encontram desligados ou sobrecarregados com muitos pedidos e processamento, a plataforma tem de implementar mecanismos de deteção de tais situações, armazenar os pedidos pendentes e ir tentando comunicar com os mesmos através de *polling*. Uma vez que se está a considerar o pior caso onde estes componentes pertencem a serviços externos confiáveis (TTPs), é assumido que estes componentes

foram instalados e implementados com os requisitos de disponibilidade e desempenho em mente. Assim, e para diminuir a quantidade de informação e tráfego de rede que chega à plataforma no caso de existir um grande número de TTPs, foi tomada a decisão da iniciativa de comunicação partir da própria plataforma IoT.

A TTP de processamento está encarregada de efetuar a decifra dos dados que lhe são enviados no momento de processamento dos dados. A TTP irá receber, a certa altura, informação do mediador contendo os dados cifrados e um código compilado juntamente com a assinatura e certificação. Como está descrito na Secção 5.5, o código contém em si embutido os certificados de chave pública necessários para a validação da assinatura e outros meta-dados. A função da TTP no momento de receção desta informação é de, simplesmente, executar o código compilado incluído na mensagem proveniente da plataforma (mais detalhes na Secção 5.5). O código compilado terá de possuir mecanismos de pedido de chaves à TTP de autorização. O seu identificador e outros dados relativos ao nome do anfitrião e porta na qual a TTP de autorização está a ouvir que foram recebidas da plataforma são incluídas num ficheiro e são acessíveis pelo código.

O código compilado é também recebido da plataforma. Assim que ele chega, é criado um ambiente de execução para que esse programa possa ser executado. A nível do sistema operativo, o mesmo tem de estar configurado para que não permita ao código a ser executado, sair do ambiente de execução ou efetuar operações não permitidas. Para tal, poderia ser utilizada uma máquina virtual (por exemplo Kernel-based Virtual Machine (KVM) [43]) Esta medida tem como objetivo adicionar reforço de segurança, uma vez que o próprio código, sendo certificado, significa que é correto e que, quando executado, não atua malignamente nem efetua operações perigosas. Para evitar a exploração de vulnerabilidades no código como *buffer overflows*, terá de existir uma restrição no tipo de linguagens suportadas.

Uma API de acesso e envio de dados para a plataforma e com uma TTP de autorização, verificação da certificação do próprio código assim que recebido e gestão dos ficheiros que serão manipulados pelo código, deverá estar presente e instalado nativamente no sistema. No entanto, esse é trabalho futuro, uma vez que, atualmente e no processo de validação da solução, essa "API" está embutida juntamente com o código certificado sendo que este flui para o próprio TTP sempre que é necessário obter acesso aos dados por parte do consumidor. No entanto, não deverá existir uma API no que toca aos procedimentos de decifra dos dados. O código a ser utilizado pelo consumidor para criar um certo nível de anonimato sobre os dados, deve incluir estas funções. Como foi mencionado anteriormente neste capítulo, os algoritmos e modos de cifra podem ser arbitrários e escolhidos, pelo que não necessitam de ser fixos. Ao permitir que o código certificado tenha o seu próprio mecanismo de decifra de dados, é

possível, no futuro, implementar um sistema onde o utilizador possa controlar qual o algoritmo e modo de cifra a ser utilizado na cifra dos seus dados produzidos.

Para além da verificação da assinatura e certificação de código, são também verificadas as assinaturas das asserções SAML que foram emitidas pelas diversas entidades (Consumidor, TTP de autorização e, opcionalmente, TTP de certificação de código - foi mencionado anteriormente que esta última TTP poderá querer introduzir certas condições de utilização juntamente com a certificação do código). Caso as assinaturas sejam verificadas com sucesso, tal significa que as mesmas foram legitimamente emitidas pelas entidades mencionadas. As respetivas chaves públicas são incluídas no código certificado e, no caso da TTP de autorização, é lhe pedida diretamente a sua chave pública numa ligação segura SSL/TLS.

Caso este componente seja fornecido por um serviço externo, terá de ser implementado um registo de utilizadores na TTP. No entanto, este serviço externo nunca poderá associar um utilizador a um determinado ambiente de execução. Ou seja, o componente não pode diferenciar a execução dos programas certificados e necessitará de efetuar este processamento numa mesma máquina ou utilizando recursos partilhados de forma anónima não diferenciando e não associando os pedidos a um determinado utilizador que esteja registado. No fundo, este serviço externo poderá funcionar como uma única máquina que possui diversos recursos alugados a clientes. Este componente terá acesso a identificadores de *gateways*/dispositivos produtores e TTPs de autorização, no entanto, este componente de processamento nunca poderá associá-los a um determinado utilizador, uma vez que, só o consumidor (fornecedor de serviços), tem acesso a este tipo de associação.

Após o processamento, todo o ambiente de execução é eliminado incluindo programas, chaves e qualquer outro elemento criado no momento de acesso e os resultados são cifrados utilizando a cifra híbrida para que o consumidor dos dados possa, posteriormente, decifrá-los e aceder, exclusivamente, aos seus resultados.

5.4.5 TTP DE CERTIFICAÇÃO DE CÓDIGO

Este componente é um novo componente introduzido na arquitetura. O seus principal objetivo é a certificação de código compilado a ser executado sobre dados cuja privacidade deve ser mantida.

A emissão de certificados e validação de código é efetuada pela TTP de certificação de código e está encarregada de efetuar toda a verificação prévia do código fonte, compilação e posterior certificação, gerando certificados, do mesmo, caso este seja válido e esteja em conformidade com as regras de privacidade dos dados e cujo nível esteja

acima de um determinado limiar definido como mínimo para que a privacidade seja respeitada e ao mesmo tempo dar utilidade ao resultado da execução deste código que irá, posteriormente, ser devolvido ao consumidor final para apresentação ou outro fim. A validação de código para que este deva estar em conformidade com as exigências no que toca ao nível de privacidade deve ser executado por seres humanos. A plataforma IoT não deve interferir neste processo sendo que a aplicação consumidora e esta TTP irão comunicar diretamente entre si para atingir o fim referido.

Como mencionado anteriormente, em trabalho futuro, esta entidade poderá emitir, também, uma asserção SAML com termos e condições de uso do código e dos dados resultantes processados pelo mesmo. Esta asserção teria de ser validada pelo componente de autorização, fornecendo ao utilizador uma breve descrição sobre o que o programa certificado faz e restrições sobre a utilização dos respetivos resultados.

5.4.6 APLICAÇÕES CONSUMIDORAS

Estas aplicações pertencem, tipicamente, a fornecedores de serviço que pretendem estudar o consumo dos clientes. Como tal, e como foi referido anteriormente, estas aplicações têm de estar relacionadas com as TTPs dos utilizadores. É de esperar que cada cliente tenha a sua própria área de configuração nesta aplicação. Ou seja, os clientes têm de efetuar um registo nesta entidade, o que já acontece normalmente.

No momento de *login*, os utilizadores devem ser capazes de configurar e associar os seus dispositivos. Esta operação desencadeia um processo de registo e associação de *hardware* que pode ser visto na Secção 5.5. Este registo é iniciado pelo próprio consumidor.

Após ser feito esse registo, a aplicação consumidora necessitará de produzir um programa que consiga aceder aos dados. Este programa terá de ser enviado para a TTP de certificação de código. Desta TTP é recebido o código assinado e certificado, caso o programa produzido seja produzido mediante as condições dessa TTP de forma a fornecer a privacidade dos dados desejada aos utilizadores. Como foi mencionado acima, este código é registado na plataforma.

Para além da informação do código a ser executado, do intervalo de tempo e da lista de dispositivos produtores/*gateways* e TTPs, o consumidor necessita de providenciar, também, uma asserção SAML. Esta asserção afirma que a TTP de processamento pode executar o código fornecido por um determinado consumidor, executando um determinado programa que irá processar dados de determinados *gateways* ou dispositivos num determinado intervalo de tempo. Esta asserção irá ter de ser validade pela TTP de processamento e é muito semelhante à asserção produzida pela TTP de autorização,

mais tarde.

5.4.7 REGISTO

Este componente introduzido é um componente de registo que tem como único objetivo armazenar associações entre *gateways* e TTPs de autorização assim como informação de contacto de várias entidades. Esta informação de contacto inclui o endereço IP e a porta em que as diferentes entidades externas estão a ouvir. As entidades aqui registadas incluem os consumidores de dados, as TTPs de autorização e os TTPs de processamento.

Sempre que qualquer componente da plataforma necessite de comunicar com algum destes componentes, é efetuada uma comunicação com o componente de registo.

Da mesma forma, quando um *gateway* comunica com a plataforma, o registo é consultado de forma a que a plataforma saiba qual a TTP de autorização associada, reencaminhando a mensagem do *gateway* para o endereço IP e porta desse TTP.

5.5 FLUXO DE INFORMAÇÃO

Nesta secção é detalhado o modo como os dados passam entre os diversos componentes e todas as operações que são efetuadas antes e depois da receção/envio de mensagens/dados por cada componente.

De notar que quando se mencionam o uso de cifras assimétricas, chaves públicas ou chaves privadas, está a ser tido em o uso do algoritmo RSA-2048 [56]. Foi considerada a hipótese de ser utilizado o algoritmo de cifra ECC [40] para proteger dados utilizando um tipo de cifra assimétrico. No entanto, este algoritmo não é muito utilizado na indústria quando comparado ao RSA, pelo que não esteve sujeito à mesma quantidade de testes e tentativas de encontro de vulnerabilidades. Para além disso, o RSA confia no problema de fatorização de inteiros para garantir segurança que tem vindo a ser estudado à muito mais tempo do que o mecanismo utilizado no ECC (logaritmo discreto em curvas elíticas). O RSA é o algoritmo de cifra assimétrica mais utilizado e mais fiável até ao momento, não tendo sido encontrada qualquer vulnerabilidade e sendo mais rápido do que o ECC. O uso de chaves de 2048 bits é propositado. Chaves de maior dimensão proporcionam uma maior segurança aplicada aos dados que necessitam de ser protegidos ao efetuar cifras ou assinaturas utilizando este algoritmo.

No contexto da IoT, todas as plataformas necessitam de suportar, pelo menos, estes dois tipos de operações: a operação de recolha e armazenamento de dados e a operação

de acesso a esses mesmos dados. Com a adição de mecanismos de segurança foi necessário alterar a forma como estas operações são processadas pelos diferentes componentes, assim como a forma como estes comunicam. Adicionalmente, foi necessário incluir operações adicionais antes que fosse possível efetuar o acesso a dados armazenados: operações de certificação de código a ser executado em TTPs de processamento de forma a que a privacidade dos dados que irá processar seja mantida e operações de registo de modo a estabelecer relações entre as entidades que o requisitam. São, então, estas quatro operações essenciais que o sistema seguro tem de suportar. Estas serão agora detalhadas nas secções seguintes.

5.5.1 OPERAÇÃO DE REGISTO

Nesta operação a aplicação procede à associação do diferente *hardware* pertencente a um utilizador assim como o registo do consumidor nos diversos TTPs e plataforma IoT.

A aplicação consumidora pode ser tratada como um fornecedor de serviços que necessitará de obter certa informação acerca da utilização dos seus serviços por parte dos clientes. Desta forma, o cliente ou utilizador em questão terá de efetuar um registo (típico nome de utilizador e palavra-passe) na entidade consumidora. Este registo é banal e não é aqui coberto.

Da mesma forma, caso o componente de autorização e processamento não sejam proprietários ao utilizador e necessitem de ser alugados a um serviço externo, então o utilizador poderá ter de se registar nesse serviço de forma semelhante ao registo mencionado antes. O utilizador necessitará, também, de configurar a TTP de autorização para que esta devolva uma lista de TTPs de processamento permitidos a serem utilizados para processamento de dados em claro.

Após o utilizador ter iniciado sessão na sua conta do seu fornecedor de serviços (consumidor de dados), este terá a opção de adicionar, registar e associar o *hardware* que possui (*gateways*, dispositivos produtores e TTPs). É este tipo de registo e associação de componentes identificados que será descrito nesta secção.

Na Figura 5.8 está representado o fluxo de dados tendo como intervenientes a aplicação consumidora, o *gateway*, a plataforma IoT, a TTP de autorização e a TTP de processamento. Os diferentes passos e operações irão ser explicados de seguida.

R1: Assim que o utilizador introduz a informação de acesso (endereço IP, porta e identificador) da TTP de autenticação e efetua a sua associação com um determinado conjunto de *gateways*, a aplicação consumidora inicia um pedido de registo no mesmo de forma direta, ou seja, sem que os dados de registo passem

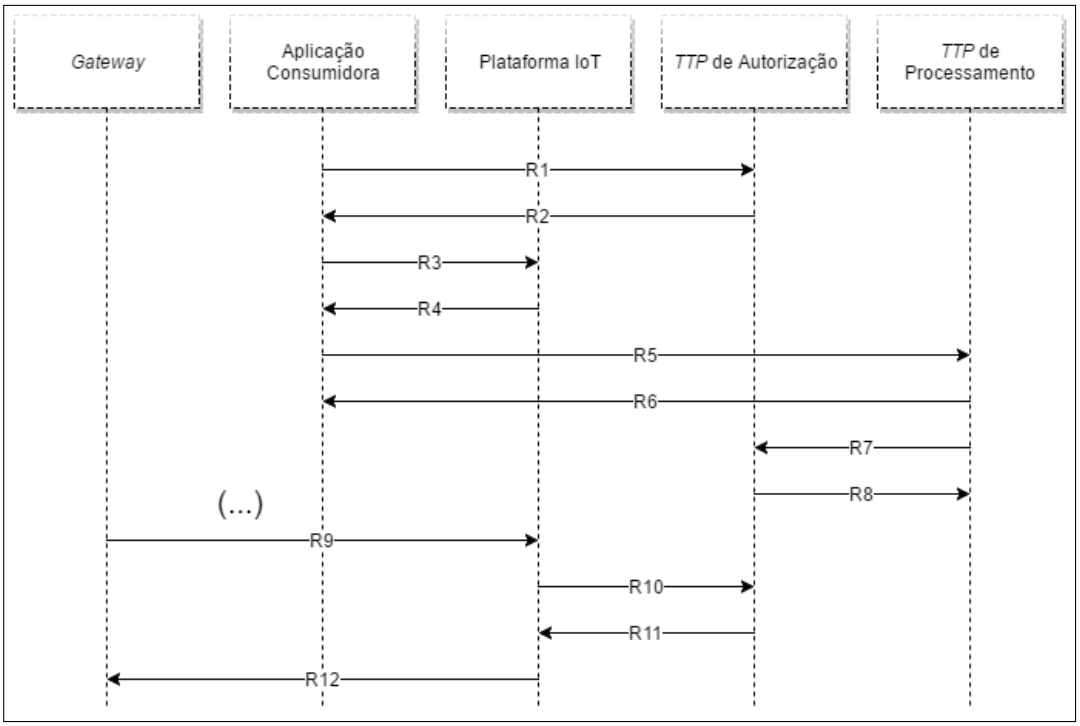

Figura 5.8: Fluxo de dados da arquitetura que se propõe na operação de registo e estabelecimento de relações entre os vários componentes/entidades

pela plataforma IoT. De notar que, como já foi referido, esta ligação é segura, uma vez que são utilizados os protocolos SSL/TLS para garantir a segurança nesta comunicação. O consumidor necessita de fornecer informação acerca dos identificadores dos *gateways* que o utilizador registou e associou a esta TTP de autorização. Adicionalmente, também pode ser aqui configurado e enviado, para cada TTP de autorização, a quantidade de minutos que são adicionados ao instante de tempo atual, aquando da geração do prazo de validade das chaves simétricas. Por exemplo, se este parâmetro for de 30 minutos. As chaves geradas pela TTP de autorização para determinado *gateway* têm um prazo de 30 minutos. No entanto, esta configuração pode ser feita manualmente, pelo utilizador, através da utilização da interface que a TTP de autorização dispõe, ao invés de intervir o consumidor (fornecedor de serviços) neste tipo de configurações.

R2: A TTP de autorização procede ao armazenamento desta informação na sua base de dados interna. Uma vez que o utilizador já necessitou de se registar e configurar a TTP de autorização previamente, é retornada a lista de TTPs de processamento permitidos. Nesta resposta, para cada elemento nesta lista, estão incluídas informações como o identificador, o endereço IP e porta da TTP de processamento.

R3: O consumidor necessita agora de enviar pedidos de registo de todas as TTPs e associações entre a TTP de autorização e os *gateways* para a plataforma IoT.

A informação a incluir é, para cada *gateway*, o identificador do *gateway* e o identificador, endereço IP e porta da respetiva TTP de autorização a ser associado a esse *gateway*. Adicionalmente, é também necessário incluir informação acerca dos TTPs de processamento que foram anteriormente permitidos. Essa informação inclui, para cada TTP de processamento, o seu identificador, o endereço IP e porta.

R4: A plataforma (nomeadamente o componente de registo) armazena a informação que recebe. Agora, sempre que a plataforma necessite de comunicar com estes componentes, já o pode fazer, uma vez que tem armazenada persistentemente os endereços IP, portas e identificadores dos *gateways* e TTPs. Na comunicação que existe entre os *gateways* e as TTPs de autorização, a plataforma também já pode reencaminhar as mensagens provenientes de ambos os lados, pois já tem também armazenadas as associações entre esses dois componentes.

R5: O próximo passo é comunicar com as TTPs de processamento que foram retornados pela TTP de autorização correspondente. Esta comunicação é feita para que, posteriormente, a TTP de processamento possa pedir à TTP de autorização correta, a sua chave pública. A informação que é enviada neste pedido é a identificação da aplicação consumidora, a identificação da TTP de autorização, o seu endereço IP e a sua porta.

R6: A TTP de processamento irá agora comunicar com esse TTP de autorização mencionado diretamente (através de uma ligação SSL/TLS segura) e pedir a sua chave pública.

R7: A TTP de autorização devolve a sua chave pública RSA em formato Privacy Enhanced Mail (PEM) [47].

R8: A TTP de processamento irá armazenar num ficheiro a chave pública recebida.

R9: Só depois de se ter efetuado este registo e que se deve proceder à inicialização do *gateway* em questão para que este se possa registar perante a TTP de autorização associado. Na mensagem a enviar para a TTP terá de comunicar a sua chave pública. Para evitar que nenhuma outra entidade (nomeadamente a plataforma) tenha alterado esta chave na sua passagem (por exemplo, através de um ataque *man-in-the-middle*), é utilizado um mecanismo que dificulta essa mesma alteração da chave pública cuja chave privada respetiva seja válida e permita ao atacante decifrar as chaves simétricas que são, posteriormente, cifradas com a chave pública em questão. Este mecanismo consiste no envio por parte do *gateway* de um conjunto aleatório de operações efetuadas sobre a chave pública legítima produzindo um código relativamente pequeno (4 caracteres). As operações que são efetuadas sobre a chave pública são funções de *hash*. É escolhido um número aleatório de *hashes* a serem efetuados sobre a chave pública. No resultado, é escolhido

um *offset* aleatório em que o resultado final é um número de *bytes* que, quando codificados em base 64, resultam num código de 4 caracteres. A chave pública, o número de *hashes* e o *offset* aleatório é enviado para a TTP.

R10: A plataforma IoT verifica, simplesmente, qual a TTP de autorização associada ao *gateway* em questão, procura pela informação do endereço IP e porta do destino e reencaminha a mensagem recebida.

R11: Ao receber a informação, a TTP efetua as mesmas operações sobre a chave pública, o que, se não houve interferência, gerará o mesmo código de 4 caracteres. O utilizador pode, então, aceitar ou não o registo deste *gateway* comparando os códigos mostrados pelo *gateway*, num ecrã OLED, por exemplo e pela TTP de autorização na sua interface Web. Caso sejam iguais, o utilizador poderá aceitar o registo do seu próprio dispositivo no momento de inicialização e garantir que não houve interferência externa. De notar que, caso o *gateway* se registe novamente com o mesmo identificador (devido a, por exemplo, ter sido reiniciado), a TTP de autorização deverá verificar que tal aconteceu e não requerer que o utilizador interaja com a TTP de forma a aceitar o pedido de registo e requisitar a verificação de códigos, novamente, sendo que esta aceitação deverá ser automática. Seguindo por este mecanismo acabado de mencionar, permite-se que a TTP de autorização tenha a certeza que está a comunicar com o *gateway* correto, ou seja, é feita uma autenticação do *gateway* na TTP de autorização. No entanto, o *gateway* não faz ideia se está ou não a comunicar com o TTP que pensa que está a comunicar. Assim, é necessário criar um mecanismo que permita autenticar a TTP no *gateway*. Uma vez que a TTP recebeu a chave pública correta e caso o utilizador autorize que o registo prossiga, a TTP continuará e gerará um código aleatório de 32 *bytes*. Este código é cifrado utilizando a chave pública do *gateway*. Este código aleatório será armazenado pela TTP.

R12: A plataforma IoT verifica, simplesmente, a qual *gateway* a TTP de autorização em questão está associado e reencaminha a mensagem recebida. Ao chegar ao *gateway*, este decifra o código aleatório e armazena-o. Agora as duas entidades possuem um segredo que só ambas conhecem. Desta forma, sempre que a TTP precisar de comunicar as chaves simétricas para cifra de dados, é utilizado o Hash-based Message Authentication Code (HMAC) [45] para juntar a esses dados um código de autenticação de mensagem (onde o parâmetro de entrada é o segredo partilhado). Posteriormente, quando a chave simétrica chegar ao *gateway*, este gera o código de autenticação de mensagem utilizando também o HMAC e o segredo partilhado. Se ambos forem iguais, então não houve interferência externa e o *gateway* pode ter a certeza de que está a comunicar com a TTP de autorização correta.

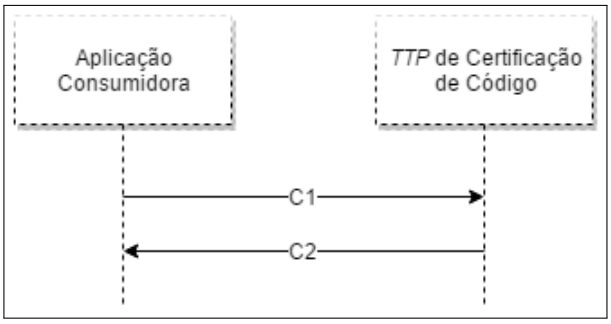

Figura 5.9: Fluxo de dados da arquitetura que se propõe na operação de certificação de código por parte do consumidor com a TTP de certificação de código

5.5.2 OPERAÇÃO DE CERTIFICAÇÃO

Nesta operação a aplicação consumidora procede à certificação do código que pretende executar nas diferentes TTPs de execução. Esta operação é simples apenas envolvendo o pedido de certificação de código por parte da aplicação, processamento do pedido incluindo a validação de código e a geração de uma resposta com o respetivo certificado que certifica a validade e execução fiável de código que não comprometa a privacidade dos utilizadores utilizando mecanismos de generalização e anonimato apropriados. Na Figura 5.9 está representado esse fluxo tendo como intervenientes a aplicação consumidora e a TTP de certificação de código. Os diferentes passos e operações irão ser explicados de seguida.

C1: A aplicação consumidora procede ao envio do código fonte a ser executado nas TTPs de processamento assim como o processo de compilação a ser efetuado e um identificador da aplicação. De notar que, embutido no código a ser aprovado, deve estar presente a chave pública do consumidor de dados;

C2: A TTP de certificação de código procede à validação do código, executada por humanos. Caso o código seja aceitável de execução para os fins mencionados nesta dissertação, a TTP procede à compilação do código, caso seja necessário, e assina com a sua chave privada interna o conjunto de código compilado. Caso este não seja um ficheiro do tipo `.jar`, o conjunto final de compilação é comprimido num ficheiro do tipo `.zip`. O ficheiro resultante será sempre um ficheiro com o formato `zip` que pode ser descompactado. A este ficheiro resultante é ainda adicionado um novo ficheiro de nome `ScotManifest.json` que irá conter a assinatura do ficheiro comprimido original, o certificado de chave pública da TTP de certificação de código adicionando a respetiva cadeia de certificados, se existente. Finalmente, são adicionados também a esse ficheiro um comando de lançamento do programa e o identificador da aplicação que efetuou o pedido. Este ficheiro é adicionado ao ficheiro de compressão e o pacote comprimido resultante é enviado de volta para a

aplicação consumidora. Opcionalmente, e como trabalho futuro, o certificador de código pode adicionar uma asserção SAML assinada, com informação relativa às condições de utilização do código certificado e de utilização dos resultados obtidos do processamento de dados utilizando este programa certificado. Estas condições poderiam estar disponíveis numa ligação externa. Esta asserção, teria de ser validada, mais tarde pelo utilizador (fazendo uso da sua TTP de autorização). Estas condições de utilização iriam influenciar a decisão do utilizador em aceitar o acesso e processamento dos seus dados ou não.

5.5.3 OPERAÇÃO DE PRODUÇÃO DE DADOS

Nesta operação o dispositivo produtor envia os seus dados criados e obtidos através do ambiente e contexto de utilização em que se encontra para que estes sejam armazenados, persistentemente, pela plataforma IoT. Na Figura 5.10 está representado o fluxo de informação que navega entre os diferentes componentes tendo como intervenientes o dispositivo produtor, o *gateway*, a plataforma IoT e a TTP de autorização. Os diferentes passos e operações irão ser explicados de seguida. A operação descrita aqui consiste em dois passos. O *gateway* é o componente responsável pela cifra dos dados provenientes dos dispositivos produtores. Esta cifra é essencial para que mais nenhuma entidade na rede possa aceder os dados confidenciais. Este componente não pode gerar a chave, porque não tem capacidades suficientes para armazenar um conjunto largo de chaves a longo prazo de forma persistente, é desenvolvido de forma a que não necessite de ter uma interface com o utilizador elaborada para que este possa gerir e ter controlo sobre todas as chaves e, finalmente, o *gateway* não pode ser contactado posteriormente de forma a que uma outra entidade possa obter as chaves em questão para efetuar a decifra dos dados. Relativamente a este último facto, vimos que os dispositivos *gateways* não têm suficientes capacidades de comunicação com a rede de forma a que tal possa ser concretizado. Como consequência, a chave simétrica de cifra dos dados terá de ser proveniente da TTP de autorização. Desta forma, os dois passos são: a obtenção de uma chave de cifra (de A1 a A5) e a cifra e envio dos dados para a plataforma (A1 e A6). O primeiro passo é efetuado apenas quando a chave simétrica de cifra não é mais válida, analisando o intervalo de tempo de validade fornecido pela TTP numa anterior consulta à TTP ou quando o *gateway* não tem em sua posse qualquer chave de cifra. Por isso, este primeiro passo não precisa de estar sempre a ser executado. Quando é necessário que o *gateway* cifre e envie dados para a plataforma e a chave simétrica de cifra ainda é válida, apenas é executado o segundo passo.

A1: O dispositivo produtor envia os seus dados, tal como os cria, para o *gateway*;

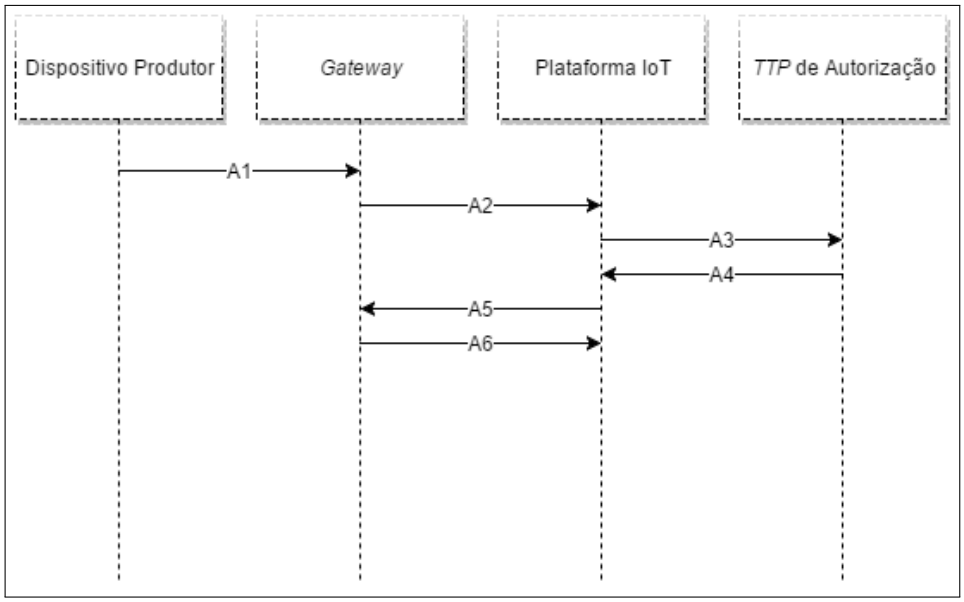

Figura 5.10: Fluxo de dados da arquitetura que se propõe na operação de produção de dados por parte dos dispositivos produtores

A2: É assumido que o *gateway* possua um par de chaves assimétricas: uma chave pública e uma chave privada previamente geradas no momento de inicialização. Na arquitetura que é proposta, os dados são cifrados com diferentes chaves simétricas. Isto, porque, utilizar sempre a mesma chave simétrica para efetuar a cifra de uma quantidade enorme de dados que são recolhidos, pode pôr em causa a segurança destes. Tal acontece porque, caso um atacante recolha muitos exemplos de informação cifrada com a mesma chave simétrica, pode proceder à utilização de técnicas de criptanálise para conseguir deduzir a chave de cifra utilizada. Estas técnicas de criptanálise são técnicas que analisam diversos exemplares de dados cifrados, comparando-os e, eventualmente encontrando semelhanças entre eles ou tentando, de alguma forma e utilizando textos cifrados cujo texto em claro seja conhecido, encontrar certas semelhanças que permitam ao atacante obter a chave simétrica. Optando pela utilização de várias chaves simétricas, fazendo a sua troca periodicamente (periodicidade esta configurada pelo utilizador na TTP de autorização controlada pelo mesmo), resulta numa menor probabilidade de um atacante poder vir a encontrar a chave secreta correta e, assim, ter acesso a todos os dados que são enviados pelo *gateway*. Caso seja a primeira vez que esteja a enviar dados protegidos ou o intervalo de tempo de validade da chave simétrica de cifra que possui tenha expirado, o *gateway* deve proceder a um pedido de uma nova chave. Para tal, envia uma mensagem de pedido de nova chave para a plataforma incluindo um identificador de pedido gerado no momento.

A3: A plataforma IoT tem informação sobre o *gateway* e com qual TTP de autorização

é que este está associado (associação esta efetuada na operação de registo cujo fluxo está mencionado na Figura 5.8) e, por isso, limita-se a reencaminhar a informação para a TTP adicionando o identificador do *gateway* em questão para que a TTP possa agir corretamente.

A4: A TTP de autorização ao receber o pedido do *gateway*, começa por gerar uma chave simétrica aleatória com o tamanho correto para ser utilizado no algoritmo de cifra AES. Mais uma vez, caso, num trabalho futuro, se implemente um mecanismo de escolha de algoritmos e modos de cifra a serem utilizados, o utilizador poderá ter controlo sobre os tipos de chaves simétricas a gerar para cada *gateway*. De seguida, a TTP acede à sua base de dados (onde se encontra armazenada a informação do *gateway* criada no momento de registo) e recupera a informação do *gateway* que pediu a chave nova. A informação recuperada, é a chave pública do *gateway* e o segredo partilhado que cujo procedimento de partilha foi mencionado no fluxo dos dados na operação de registo, de forma a que a TTP se autentique perante o *gateway*. Se o *gateway* não foi previamente registado, é devolvida uma mensagem de erro de volta para o *gateway* ao invés da nova chave. De seguida, a TTP consulta as suas configurações do utilizador referente ao *gateway* de forma a obter informação sobre o prazo de validade que deve ser estabelecido para esta nova chave. Após isso, são criados dois instantes de tempo: ambos indicam instantes de tempo, de início e de fim (semelhante ao que existe nos certificados X.509 [35] da infraestrutura de chaves públicas já existente (PKI)), criando um intervalo de tempo em que o uso da chave simétrica gerada é válida para cifrar os dados. De forma a lidar com casos em que os diversos relógios dos componentes não estejam sincronizados, um intervalo de tempo pequeno é adicionado aos limites de validade. Este pequeno intervalo pode também ser configurado pelo utilizador. Assim, é natural que algumas chaves possuam sobreposições de prazos de validade. De seguida, a TTP armazena a chave simétrica aleatória criada internamente assim como o intervalo de tempo correspondente ao prazo de validade. Esta informação será necessária mais tarde quando for necessário efetuar a decifra de dados pela TTP de processamento. Depois, sobre o conjunto: chave simétrica gerada e os dois instantes de tempo que definem a validade da chave, é aplicado o HMAC utilizando como modo de *digest* o algoritmo SHA-256 e utilizando o segredo partilhado recuperado da sua base de dados. É gerado um código de autenticação de mensagens. Foi este o algoritmo utilizado em todas as operações de *hashing* na arquitetura que é aqui proposta. A versão 2 do SHA (que é a utilizada) ainda não se encontra vulnerável no estado da arte atual sendo, por isso, uma algoritmo seguro. Adicionalmente, o resultado tem um tamanho de 256 *bits*. Este tamanho é grande o que faz com que a existência de colisões seja

suficientemente pequena de forma a não comprometer a segurança dos *digests* gerados em qualquer parte da arquitetura proposta. O código de autenticação de mensagens gerado é concatenado à própria chave (e validade) e este conjunto é cifrado com a chave pública do *gateway* que foi comunicada à TTP de autenticação na fase de registo. O identificador da TTP e o conjunto cifrado é enviado para a plataforma com o intuito de chegar ao *gateway*. A esta resposta, está também incluído o identificador de pedido, recebido no pedido da chave.

A5: A plataforma IoT tem informação sobre a TTP e com qual *gateway* é que este está associado (associação esta efetuada na operação de registo cujo fluxo está mencionado na Figura 5.8) e, por isso, limita-se a reencaminhar a informação para o *gateway*.

A6: Caso o *gateway* tenha tido a necessidade de pedir uma nova chave, este componente precisa ainda de efetuar algumas operações antes de poder prosseguir com a cifra dos dados e envio para a plataforma IoT. Essas operações começam por decifrar a informação recebida pela TTP de autorização utilizando a chave privada do *gateway*. De seguida, o *gateway* procede com a separação do código de autenticação da mensagem com os restantes dados. O mesmo processo da aplicação do HMAC utilizando o segredo partilhado também armazenado pelo *gateway* na fase de registo é aplicado sobre os dados e são comparados os códigos de autenticação de mensagens. Se forem diferentes significa que os dados não são fidedignos ou sofreram alterações ao nível de integridade durante a comunicação. Caso sejam válidos, o *gateway* armazena a chave simétrica e a validade e elimina a informação anterior (caso exista), pois agora é esta a chave simétrica em vigor durante o seu prazo de validade. Agora o *gateway* está pronto para cifrar os dados e enviá-los para a plataforma de forma a serem armazenados na base de dados de forma persistente. O *gateway* começa por criar uma *hash* criptográfica dos dados que recebe, utilizando o algoritmo de *hashing* SHA-256. O resultado é concatenado com os dados originais recebidos. Todo este conjunto necessita de ser cifrado com a chave simétrica, ainda válida, armazenada e recebida anteriormente utilizando o algoritmo criptográfico AES, no modo CFB. Para efetuar esta cifra, o *gateway* necessita de criar um IV que é acompanhado com os dados cifrados. Este IV deverá ser aleatório e gerado sempre que se pretende cifrar novos dados provenientes de sensores ou dispositivos produtores. Uma vez que o *digest* resultante da aplicação do *hash* são 256 bits de saída, é possível, no momento da decifra de todo o conjunto efetuado mais tarde pela TTP de processamento, separar o *digest* dos dados recebidos pelos dispositivos produtores. Isto é feito para que seja possível manter a integridade dos dados, onde quer que estes sejam mais tarde armazenados. O *gateway* não armazena informação em bruto e não necessita, por isso, de uma

base de dados, mantendo o *gateway* um componente simples em todo o sistema adicionando apenas capacidade de cifra. A seguinte informação é enviada para a plataforma com o objetivo de ser, posteriormente, armazenada persistentemente: instante de tempo de cifra dos dados (útil para quando for necessário recuperar a chave simétrica correta da TTP de autorização), o IV utilizado e requerido pelo modo de cifra CFB e os dados cifrados.

5.5.4 OPERAÇÃO DE CONSUMO DE DADOS

Nesta operação a aplicação consumidora pretende aceder aos dados que foram produzidos por determinados dispositivos. No entanto, a revelação dos dados originais poderia pôr em causa a privacidade dos utilizadores donos dos dados e dos dispositivos que os produziram. Deste modo, é aqui aplicado o mecanismo, referido anteriormente, de acesso a uma transformação/vista dos dados (uma forma generalizada dos mesmos). Na Figura 5.11 está representado o fluxo de informação entre os diversos componentes para que o acesso com preservação de privacidade possa ser efetuado, tendo como intervenientes a aplicação consumidora, a plataforma IoT, a TTP de autorização e a TTP de processamento. Os diferentes passos e operações irão ser explicados de seguida.

É importante notar que, após toda a fase de registo (Figura 5.8) e antes de qualquer operação de consumo de dados, é necessário que o próprio consumidor forneça à plataforma certas informações de contacto. Para tal, o consumidor começa por enviar para a plataforma o seu identificador, o seu nome, o seu endereço IP e a sua porta. Assim, depois quando enviar o pedido à plataforma para consumir dados, a plataforma saberá como iniciar uma comunicação com o consumidor. Agora, é necessário efetuar o registo do código certificado na plataforma. Este registo é simples e existe com um único propósito. O de evitar que sempre que o consumidor necessite de aceder a dados sob proteção de privacidade, tenha de enviar todo o conjunto (*bundle*) de código para a plataforma. Assim, sempre que o consumidor precise de utilizar um novo código certificado de processamento, este apenas o tem que enviar, juntamente com a sua identificação, para a plataforma. Esta tratará de o armazenar, criar um identificador do código em questão e retorná-lo, como resposta, ao consumidor. O consumidor terá, obviamente, de armazenar este identificador para poder utilizá-lo mais tarde. Acabados estes procedimentos, o consumidor fica em condições de consumir dados.

1: A aplicação consumidora de dados pretende aceder aos dados produzidos por um determinado conjunto de dispositivos num determinado intervalo de tempo. Para tal, a aplicação efetua um pedido à plataforma IoT (utilizando a camada de exposição de serviços (API)) contendo a seguinte informação:

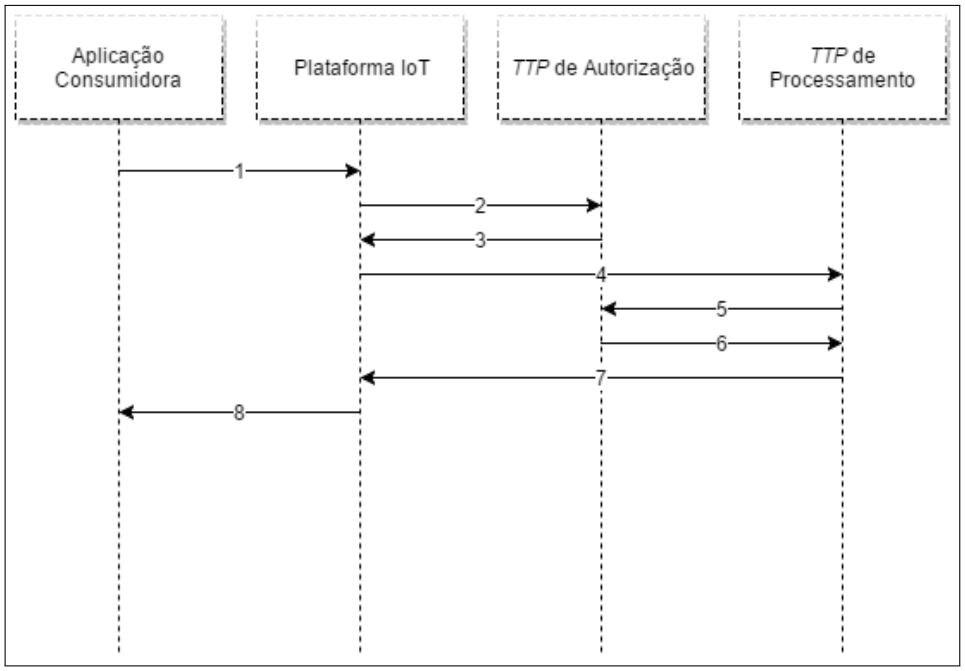

Figura 5.11: Fluxo de dados da arquitetura que se propõe na operação de acesso aos dados parte do consumidor

a) Identificador do pedido gerado no momento;
b) Identificador do consumidor;
c) Identificador do código compilado a ser executado sobre os dados em claro, na TTP de processamento;
d) Lista de identificadores dos dispositivos produtores ou *gateways* que o consumidor pretende aceder para obter dados;
e) Um intervalo de tempo constituído por dois instantes de tempo de início e de fim, que limitam a quantidade de dados produzidos a serem retornados, pois apenas os dados criados cujo instante de criação esteja contido no intervalo de tempo especificado é que são incluídos na resposta;
f) Identificador da TTP de autorização que está associada aos *gateways* (informação que o consumidor conhece);
g) Identificador de uma TTP de processamento que tenha sido permitida pela TTP de autorização na fase de registo;
h) Certas condições a serem respeitadas (como por exemplo o tempo máximo que a TTP de processamento pode demorar para processar os dados e retornar os resultados).
i) Asserção SAML assinada que indica que este consumidor pretende aceder a dados de uma lista de *gateways* num determinado intervalo de tempo executando um determinado código (identificado pelo seu *digest* no ponto de vista do utilizador) onde será executado numa determinada TTP de

processamento. A informação enviada referida acima, seria incluída nesta asserção assinada como condições de utilização a serem verificadas pela TTP de processamento, mais tarde.

2: A plataforma acede à base de dados através do componente de armazenamento e usa a lista de identificadores e o intervalo de tempo que recebeu para construir uma pesquisa de acesso ao armazenamento. Esta pesquisa irá resultar no retorno dos dados cifrados, armazenados persistentemente nas diversas operações de produção de dados, efetuadas pelos vários dispositivos produtores ou *gateways*. Os dados obtidos são retornados do armazenamento. A plataforma prossegue à obtenção do código certificado e do comando de lançamento do programa utilizando o identificador recebido do consumidor. A plataforma necessita, agora, de comunicar com a TTP de autorização associada aos *gateways* de forma a obter a autorização expressa do utilizador dono dos dados obtidos do armazenamento na sua forma cifrada. Para tal, a plataforma envia para a TTP de autorização a informação recebida do consumidor (identificadores da TTP de processamento, consumidor, pedido, lista de dispositivos, impressão digital do código certificado (*digest*) etc. à exceção da asserção SAML) e incluindo um pedido de autorização SAML que será emitido pela TTP de autorização.

3: A TTP de autorização procede à apresentação de toda esta informação ao utilizador através de uma interface, por exemplo, Web. O utilizador terá de autorizar ou não o acesso aos seus dados por parte do consumidor identificado acedendo a uma secção de ações pendentes na TTP de autorização. O código certificado que irá ser utilizado não é aqui especificado e é, antes disso, especificada uma impressão digital do mesmo. Desta forma, o utilizador terá de, por outros meios, verificar o código em causa e tomar a decisão de autorização. Este passo da verificação de código pode não ser necessário uma vez que o código é certificado por uma entidade confiável que garante que os parâmetros de manutenção de privacidade estão bem aplicados. No entanto, caso o utilizador tenha conhecimentos sobre programação, o mesmo poderá, por exemplo, aceder à interface Web do consumidor e verificar o código identificado pela sua impressão digital. Caso o utilizador autorize o acesso aos dados, a TTP de autorização procede à elaboração de uma asserção SAML assinada que contém condições e informação semelhante à emitida pelo consumidor em um dos passos anteriores. A asserção gerada e o mesmo identificador de pedido recebido no pedido são enviados para a plataforma.

4: A plataforma, ao receber a asserção gerada pela TTP de autorização fica a ter em sua posse duas asserções SAML. O próximo passo é enviar o código certificado, o comando para o seu lançamento, o identificador da TTP de autorização utilizado e associado ao utilizador dono dos dados, o identificador do consumidor,

o identificador do pedido, os dados obtidos do armazenamento persistente e as duas asserções SAML emitidas pela TTP de autorização e pelo consumidor para a TTP de processamento indicado pelo consumidor.

5: Toda esta informação chega, então, à TTP de processamento responsável pela execução do código certificado. Neste momento, e como já foi referido anteriormente, a TTP de processamento não tem, por si só, uma API a ser utilizada pelo programa certificado, nativa no sistema. Tal seria feito em trabalho futuro pelo que, de momento, o programa certificado terá de efetuar todas as validações e verificações de assinaturas e asserções SAML. No entanto, não é grave, em termos de segurança em ser o código o responsável por tal, uma vez que o código é certificado previamente por uma entidade que verifica o seu correto funcionamento. No entanto, em termos de usabilidade, tal abordagem não é tão boa uma vez que exige que o consumidor efetue todas essas operações e conheça todos esse mecanismos de segurança. Este facto também não facilita a verificação e validação de programas por parte da TTP de certificação de código. A TTP começa por efetuar a preparação do ambiente de execução, criando um diretório para execução do código. Toda a informação recebida no pedido da plataforma, será armazenada nesse diretório para que o código possa ter acesso. Será um ficheiro JSON de nome `MetaData.json` onde o pedido, no formato JSON, recebido é literalmente armazenado num ficheiro para que seja acessível pelo código certificado. No entanto os dados recebidos irão para um ficheiro diferente: `Data.json`. O código certificado é colocado também no diretório e é extraído e eliminado da pasta comprimida (*bundle*) o ficheiro `ScotManifest.json`. O código certificado está agora no seu estado original e pronto para que a sua assinatura (presente no manifesto) seja validada. O programa certificado deverá, também, poder ter acesso às diferentes chaves públicas da TTP de processamento assim como certificados confiáveis, nomeadamente o certificado confiável da TTP de certificação de código. O programa certificado é agora posto em execução utilizando um comando de sistema dependendo da linguagem utilizada. Este programa necessitará de validar o certificado de chave pública e respetiva cadeia de certificados (opcionalmente) da TTP de certificação de código presente no manifesto. Para tal, a TTP de processamento deverá aceder à lista de certificados confiáveis. O programa deverá proceder também à verificação da assinatura da asserção SAML emitida pelo consumidor utilizando a chave pública do consumidor incluída no ficheiro comprimido correspondente ao código. O programa deverá proceder também à verificação da assinatura da asserção SAML emitida pela TTP de autorização utilizando a chave pública da TTP obtida previamente no processo de registo. De seguida é executado um passo importante. As duas asser-

ções SAML emitidas pelas diferentes entidades têm que estar em conformidade e não existir qualquer tipo de inconsistências no pedido que foi efetuado pelo consumidor e na autorização por ele obtida. Caso as asserções não estejam em conformidade uma com a outra, o processamento não pode continuar. De seguida o programa acede aos dados, aos instantes de tempo em que os dados foram cifrados e à informação dos *gateways*. De notar que nem todos os dados estão cifrados. Meta-dados foram adicionados pelo componente de enriquecimento já existente e não alterado da plataforma IoT (Secção 2.5). Os identificadores dos *gateways* são um tipo de meta-dados não cifrados, mas não são associados aos próprios clientes, pois a identificação do cliente e sua associação com os *gateways* só está disponível no consumidor dos dados e na TTP de autorização. Desta forma, a TTP de processamento que tem acesso aos dados em claro não poderá deduzir nada sobre uma potencial associação de identificação com um determinado cliente. Adicionalmente estes identificadores, como mencionado anteriormente, podem ser modificados e, pode, inclusive, ser implementado um mecanismo de rotação de identificadores dos *gateways* onde um único *gateway* poderia utilizar um identificador diferente sempre que enviasse novos dados. Neste caso, o utilizador, em vez de registar no consumidor o identificador para um *gateway* necessitaria de registar uma lista de identificadores. Uma vez obtida a identificação dos *gateways*, a TTP de processamento necessita de efetuar um pedido diretamente à TTP de autorização de forma a obter as chaves de cifra dos dados com o objetivo final de efetuar a sua decifra para os poder processar de seguida. Para tal é construído um pedido com a asserção SAML emitida pela TTP de autorização e com lista de *gateways* associados aos instantes de tempo de cifra que são necessários para obter as chaves corretas. Ou seja, para cada *gateway*, existirá uma lista de instantes de tempo de cifra que foram retirados de cada conjunto de dados.

6: A TTP de autorização acede à sua base de dados e, de acordo com a informação recebida, retorna à TTP de processamento, para cada *gateway*, uma lista de tuplos contendo a chave simétrica utilizada e o prazo de validade da mesma.

7: A TTP de processamento tendo em sua posse as chaves de cifra simétricas utilizadas sobre os dados, procede à sua decifra com a ajuda dos IVs (caso um modo de cifra que o requeira tenha sido utilizado) localizados em claro juntamente com os dados cifrados e com o instante de tempo de cifra. De seguida, o *digest* de 256 *bits* é separado dos restantes dados e é validado. Caso, para algum conjunto de dados, o *digest* não seja válido, significa que foi danificado. Cabe ao programa certificado decidir o que fazer nestas situações. Caso o *digest* seja válido, prossegue-se com o processamento dos dados em claro preservando a privacidade do dono dos mesmos. É possível que devido a diferentes sincronizações de relógio dos vários

componentes ou a um pedido expresso de uma nova chave de cifra ao *gateway* tenha sido feita pelo utilizador, que o prazo de validade das cifras se sobreponha. Nestes casos, são tentadas todas as chaves simétricas apropriadas ao instante de tempo de cifra cuja validade está sobreposta. Os dados serão decifrados por chaves não corretas pelo que será produzido lixo. O *digest* é aqui usado para verificar se o resultado da decifra é lixo ou são dados reais. Os resultados finais são cifrados de forma híbrida como mencionado na utilizando uma chave simétrica aleatória gerada no momento e utilizando AES no modo CFB para efetuar a cifra. A chave simétrica é depois cifrada utilizando a chave pública do consumidor presente na pasta comprimida do programa certificado. Assim, os resultados finais cifrados, os identificadores de pedido e do consumidor, a chave simétrica cifrada e o IV também gerado aleatoriamente no momento, são enviados para a plataforma.

8: A plataforma trata apenas de reencaminhar os resultados para o consumidor. O consumidor decifra a chave simétrica utilizando a sua chave privada, decifra os resultados finais utilizando a chave simétrica decifrada e obtém os resultados finais em claro. Estes últimos que não põem em causa a privacidade dos utilizadores donos dos dados produzidos. Um dos objetivos centrais desta dissertação.

As quatro operações (registo de componentes, certificação de código, produção e consumo de dados) que foram descritas detalhadamente são as principais operações da arquitetura de segurança de dados para plataformas IoT que foi proposta.

capítulo 6

Avaliação e Resultados

Neste capítulo são apresentados os resultados da arquitetura que é proposta nesta dissertação.
Primeiramente é analisado o impacto que os dados cifrados têm nas bases de dados, de forma relativa à arquitetura inicial sem mecanismos de segurança. De seguida é analisado e mostrado o impacto no desempenho também relativamente a uma arquitetura IoT não segura. São apresentados, depois, factos acerca da usabilidade do trabalho que foi efetuado. Por último, são referidos alguns aspetos sobre o tipo de informação a que cada entidade relevante na arquitetura tem acesso.

Neste capítulo são apresentados os resultados da implementação da arquitetura segura para uma plataforma IoT que é proposta nesta dissertação.

Neste capítulo recorremos a um caso de uso de medição de consumo elétrico de uma residência vulgar. Assim, é simulada a existência de um contador de eletricidade inteligente que se encontra instalado numa residência. Nesta simulação, os dados sobre o consumo de eletricidade foram enviados minuto a minuto, durante 1 mês (o mês de janeiro de 2007). Estes dados são reais e fora retirados de um repositório online[1] de aprendizagem automática da Universidade da Califórnia Irvine. Foram efetuados envios dos dados em claro e cifrados (utilizando os mecanismos de segurança da arquitetura proposta) de forma a que fosse possível efetuar a comparação a nível de tamanho necessário para o seu armazenamento em bases de dados da plataforma IoT e a nível de desempenho no acesso aos dados a partir de um consumidor. No total, foram enviados, armazenados na plataforma e acedidos um total de 44640 entradas.

[1] https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

De forma a efetuar uma análise na usabilidade, foi utilizado e apresentado um simples programa a ser executado de forma a generalizar os dados e torná-los anónimos para o consumidor. Este programa é, como foi referido na proposta de arquitetura, executado na TTP de processamento.

6.1 IMPACTO NO ARMAZENAMENTO

Os dados necessitam de ser armazenados na base de dados da plataforma IoT. Na plataforma IoT convencional, sem segurança (Secção 2.5), o tamanho dos dados armazenados relativamente aos tamanho dos dados produzidos é grande e um tanto variável dependendo do tópico utilizado e dependendo também de outros elementos como *tags* que podem ser adicionadas. Deste modo, não será aqui comparado o tamanho dos dados produzidos com o tamanho dos dados armazenados em ambas as arquiteturas (a plataforma original e a plataforma que implementa mecanismos de segurança dos dados). De notar que, juntamente com os dados produzidos, têm de ser adicionados certos meta-dados para que seja possível fazer a recuperação dos dados novamente quando for necessário aceder-lhes. Esta adição de meta-dados é efetuada pelo componente de enriquecimento da plataforma IoT e é a causa da diferença de tamanhos que existe entre o tamanho na produção de dados e o tamanho no seu armazenamento.

Nesta secção, para um determinado tamanho fixo de dados a serem armazenados nas bases de dados da plataforma, será feita uma comparação entre o tamanho que é realmente ocupado nas bases de dados utilizando a arquitetura da plataforma IoT origem e o tamanho que é realmente ocupado nas bases de dados utilizando a arquitetura que implementa mecanismos de segurança, apresentada nesta dissertação.

Na Figura 6.1 está representada uma amostra singular de informação. Estes dados são relativos a consumo de energia elétrica. É utilizado o formato JSON em que o único atributo contém valores separados por vírgulas. Estes valores são, da esquerda para a direita: a data da medida no formato `dd/mm/yyyy`; a hora no formato `hh:mm:ss`; a potência média ativa global da residência (em quilowatts); a potência média reativa global da residência (em quilowatts); a voltagem média (em volts); a intensidade de corrente média global da residência (em amperes); uma sub-medida de energia ativa (em watt-hora) numa cozinha; uma sub-medida de energia ativa (em watt-hora) numa lavandaria; uma sub-medida de energia ativa (em watt-hora) de um aquecedor de água e ar condicionado;

Estes foram os dados produzidos. O seu tamanho é de 74 caracteres o que equivale a 74 *bytes*. Mais tarde será generalizado e considerado x referente ao tamanho dos dados originais. O componente de enriquecimento da plataforma IoT adiciona os mesmos

```
{"data": "31/1/2007;23:59:00;0.326;0.126;242.800;1.400;0.000;0.000;0.000"}
```

Figura 6.1: Exemplo de uma entrada do repositório utilizado contendo informação acerca do consumo elétrico num instante de tempo

meta-dados no armazenamento em ambas as arquiteturas, pelo que não será considerada a sua existência neste cálculo.

Os dados que são de interesse (não incluindo meta-dados) armazenados nas bases de dados da plataforma utilizando a arquitetura original convencional são os referidos acima. Ou seja, são armazenados 74 *bytes* para esta amostra de dados.

Na arquitetura segura que é proposta, os dados originais serão cifrados. Para além disso terão de incluir mecanismos que permitam a verificação de integridade como foi mencionado no capítulo de descrição da arquitetura. Esta verificação de integridade passa por aplicar um algoritmo de *hashing* sobre os dados. Uma vez que o algoritmo utilizado é o SHA-256, o *digest* resultante consiste em 32 *bytes* (ou 256 *bits*). Foi necessário recorrer à conversão destes *bytes* para base64. Desta forma, os *bytes* são transformados em caracteres visíveis para efeitos de registo das operações *logging* e concatenáveis com os dados originais. A conversão para base64 de n *bytes* resulta na criação de $\left\lceil \frac{4}{3}n \right\rceil$ caracteres (ou *bytes*). A este resultado é adicionado o *padding* necessário de forma a obter um resultado final múltiplo de 4. Tendo em conta o nosso exemplo, quando o *digest* de 32 *bytes* é convertido para base64, o resultado tem um tamanho de 43 *bytes*. 44, adicionando o respetivo *padding*. Estes 44 *bytes* são então concatenados com os dados originais (74 *bytes*) sendo que a informação pronta a ser cifrada tem um tamanho de 118 *bytes*. Agora, os dados são cifrados, pelo que o resultado da cifra produz dados com tamanho de 120 *bytes*, uma vez que a cifra simétrica por blocos necessita de adicionar *padding* ao último bloco da máquina de cifra.

Generalizando, para quaisquer dados de tamanho x *bytes*, o resultado da cifra é, em *bytes*, de

$$y = 4 \left\lceil \frac{(x + 44)}{4} \right\rceil . \tag{6.1}$$

O resultado desta cifra irá ser convertido, novamente e utilizando a mesma fórmula, em base64, pelo que o resultado serão 160 *bytes* para o nosso exemplo (não foi necessário adicionar *padding* adicional, neste caso). A estes 160 *bytes*, são adicionados 65 *bytes* fixos. Estes *bytes* adicionais fixos correspondem à inclusão de informação adicional juntamente com a cifra (informação esta que não necessitaria de ser adicionada na arquitetura de uma plataforma IoT convencional, pelo que estes meta-dados relativos à

cifra terão de ser tidos em conta). Esta informação contém o IV utilizado na cifra (tem sempre o mesmo tamanho) e um instante de tempo de cifra (que irá ter sempre o mesmo tamanho durante muitos anos). Os dados cifrados, o IV e o instante de tempo de cifra são encapsulados no formato JSON, uma vez que é o formato aceite pela plataforma. O tamanho resultante referido de 65 *bytes* fixos já contemplam este facto. De notar que, caso o JSON seja formado de forma diferente (diferentes chaves, por exemplo), este tamanho pode variar. Assim, para este exemplo e na arquitetura proposta, o tamanho resultante é de 225 *bytes* contra os 74 *bytes*, no caso da arquitetura convencional. Para este exemplo, existe um aumento de, aproximadamente, 204% no tamanho armazenado.

Novamente generalizando, para quaisquer dados de tamanho x *bytes* (tamanho dos dados no caso de utilização de uma plataforma IoT convencional), o correspondente tamanho adicional na arquitetura proposta é de $4\left\lceil\frac{y}{3}\right\rceil + 65$ *bytes* ou, substituindo y e sendo w o resultado final em *bytes*:

$$w = 4\left\lceil\frac{4\left\lceil\frac{x+44}{4}\right\rceil}{3}\right\rceil + 65 \tag{6.2}$$

Ainda na generalização, caso se pretenda obter o respetivo aumento p em percentagem (%) na comparação com a arquitetura de uma plataforma IoT convencional:

$$p = \frac{100\left(4\left\lceil\frac{4\left\lceil\frac{x+44}{4}\right\rceil}{3}\right\rceil + 65 - x\right)}{x} \tag{6.3}$$

Uma vez que existe um tamanho fixo base, no caso do tamanho dos dados (x) ser pequeno, o aumento, em percentagem, quando comparado com uma arquitetura convencional não segura, é extremamente grande. No caso em que o tamanho dos dados é muito grande, basta calcular o limite de quando x tende para o infinito da Equação 6.3. O limite é de $33,3\%$, aproximadamente, de aumento no tamanho de armazenamento na plataforma IoT.

De notar que, quer os dados cifrados, quer os dados em claro podem ser armazenados na mesma base de dados sendo que não existem restrições para tal.

Para concluir esta secção, é apresentado um exemplo de dados cifrados correspondentes aos dados da amostra exemplo apresentados anteriormente, juntamente com os meta-dados de cifra respetivos na Figura 6.2

Note-se que os valores mencionados não correspondem ao espaço que é utilizado pelos dados no armazenamento, pois estes dados serão ainda sujeitos a um processo de enriquecimento adicional, comum às duas arquiteturas. Para além disso, está-se apenas a analisar o tamanho dos dados que são enviados para serem, efetivamente, armazenados. O modo como o formato JSON é armazenado pela plataforma, atualmente, pode ser diferente. Era de esperar que, na aplicação de mecanismos de cifra juntamente com

```
{"cTs": 1465009931, "data":
"5ImeWyCDB+jDvHeKrdZDyJ9N4deaXuSY+GMUGOI4E4QwMCZ+ycFVKXjukVqED3k78CLkE2u6Eg
suK3sK4WAT+bQ5lB42l5h3r9eGdjSRxEE7p/BLtmoHjxowgVPNSUC5MTUy0gloUBKfVuIllst7S
VyzHEHnmQ==", "iv": "SNu+C2YVHRhkxpotMKSwZA=="}
```


Figura 6.2: Resultado da cifra de exemplo de uma entrada do repositório utilizado com a adição dos meta-dados necessários

mecanismos de integridade, o tamanho dos dados a enviar aumentasse. Não se pode fazer nada relativamente a este facto a não ser escolher, eventualmente, um mecanismo de cifra que não precise de IV, por exemplo. No entanto, não é possível escapar a um aumento considerável de dados quando são aplicados mecanismos de segurança sobre estes.

De notar também que o tamanho dos dados produzidos pelos dispositivos produtores não é alterado na arquitetura que é proposta. O tamanho dos pacotes enviados para um *gateway*, utilizando tecnologias como o WiFi [14], Zigbee [42] ou Bluetooth [29], não é alterado relativamente à arquitetura de uma plataforma IoT convencional sem os mecanismos de segurança propostos aplicados. Isto acontece porque o funcionamento dos dispositivos não é alterado para além da inclusão de mecanismos SSL/TLS ponto-a-ponto.

6.2 IMPACTO NO DESEMPENHO

Todas as arquiteturas que procuram resolver problemas de segurança, quando comparadas com as que não o procuram fazer, acabam por ser penalizadas em termos de desempenho. Este decréscimo no desempenho, ocorre devido a diversos fatores. Um dos fatores importantes está na quantidade de informação que tem que navegar, adicionalmente, de entidade para entidade. Na arquitetura convencional, para obter os dados desejados apenas era necessário efetuar um pedido a um dispositivo que expunha uma interface de acesso à base de dados. Assim que consultasse a base de dados, os dados desejados seriam retornados diretamente ao consumidor. A obtenção de dados era, por isso, simples onde apenas existia um pedido e uma resposta. Contudo, na arquitetura aqui proposta, o processo de obtenção de dados é um tanto complexo e demoroso. Como tal, a sua aplicação em aplicações em tempo real não é possível. Esta complexidade advém do facto de que, inicialmente, tem de ser feito um pedido a uma entidade externa (a TTP de autorização). Só depois do utilizador ter acesso ao

seu componente de autorização para, de facto, autorizar ou não o pedido de acesso a dados, é que o processo continua. Após esta autorização, ainda são efetuadas trocas de dados e código certificado com a TTP de processamento, sendo que esta irá novamente comunicar com a TTP de autorização diretamente de modo a obter as chaves necessárias utilizando um canal seguro. Só depois de serem processados, os resultados seguem para a plataforma, que tratará de comunicar com o consumidor de forma a enviar-lhe os resultados. Portanto, só nas diversas comunicações e trocas de dados existentes, o desempenho é logo afetado. De notar que, a qualidade e velocidade da rede pode variar de momento para momento, sendo que os valores dos atrasos podem variar e depender muito da condição atual da rede, em situações reais.

Outro fator existente que faz com que o desempenho seja afetado é o facto de se ter de efetuar a decifra de todos os dados que são recebidos. Uma vez que estes estão cifrados quase desde o momento de produção, sempre que se necessita de lhes aceder, métodos criptográficos são necessários para efetuar a decifra utilizando um algoritmo simétrico. Estes métodos podem ser potencialmente morosos dependendo da quantidade de dados desejada assim como do tamanho que cada unidade de dados possui.

Por fim, existe ainda um fator adicional que prejudica o desempenho. Este fator é o processamento efetuado sobre os dados em claro (após a decifra) de forma a obter resultados generalizados, transformados e anónimos. Dependendo, obviamente, da quantidade de dados a serem analisados e do código certificado que iá efetuar o processamento, este desempenho varia muito e não é possível obter valores aproximados generalizados dada a variabilidade que pode existir.

Recorrendo ao exemplo referido na secção anterior (Secção 6.1), foram efetuadas algumas medições de desempenho utilizando uma arquitetura para uma plataforma IoT convencional e sem ter em consideração a segurança dos dados produzidos pelos utilizadores. Todas as medições foram feitas através do cálculo da diferença existente entre tempos antes de a operação iniciar e depois da operação terminar ao nível do código. Para efetuar estas medições foi utilizada uma máquina virtual gerida por OpenStack. O sistema computacional onde esta máquina virtual está a executar possui 2GB de memória RAM, 25GB de armazenamento em disco, tem instalado o sistema operacional Ubuntu 14.04.4 LTS e possui um processador de nome Intel Xeon E312xx (Sandy Bridge) de, aproximadamente, 1.9GHz. Mais à frente são apresentados resultados relativos ao tempo despendido pelo processador em efetuar a decifra de todos os dados que pretende analisar. Uma vez que foi utilizado o algoritmo AES com chaves de 256 bits para tal, torna-se relevante mencionar que foi utilizada a biblioteca "pycrypto" que utiliza código C nativo para efetuar as operações de decifra. Adicionalmente, é também relevante mencionar que o processador utilizado não suporta Advanced Encryption Standard New Instructions (AES-NI). Ou seja, para efetuar a decifra de todos os dados

Tabela 6.1: Apresentação de medidas de desempenho medidos no acesso aos dados utilizando uma arquitetura convencional

Medidas de Desempenho - Arquitetura Convencional					
	1	**2**	**3**	**4**	**5**
Tempos (em segundos)	18,867	16,989	20,658	16,121	17,122
Média Aritmética	17,951				
Desvio Padrão	1,812				

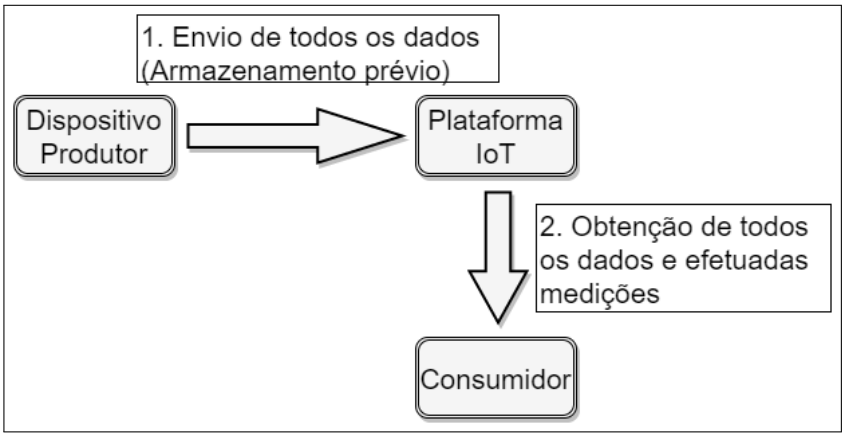

Figura 6.3: Diagrama ilustrativo dos passos efetuados no ponto de vista do consumidor para realizar medições

não foi utilizada aceleração de *hardware*. Todos os componentes da solução foram executados na mesma máquina, com exceção dos componentes que não foram alterados e que já existiam antes da adição de mecanismos de segurança a uma arquitetura convencional, como é o caso do componente de armazenamento, *brokers* de mensagens (barramento de mensagens e domínio de rede IoT) e o componente de enriquecimento.

Na Tabela 6.1, estão apresentadas as medidas de desempenho utilizando uma arquitetura para plataformas IoT convencional e na Figura 6.3. Como é possível verificar e como era de esperar, estes valores representam boas medidas de desempenho do sistema, tendo em conta o grande conjunto de dados que foi requisitado pelo consumidor. De notar que aqui apenas se mediu o tempo de recuperação da informação. Após essa recuperação não existiu qualquer processamento adicional sobre estes dados nem foram utilizados quaisquer métodos criptográficos. A única operação efetuada nesta experiência para retirar medições foi uma chamada dum consumidor de dados à API disponibilizada pelo componente de armazenamento da plataforma. Nesta chamada foi incluído um intervalo de tempo que limitou a obtenção de todo o conjunto de dados que referi no início deste capítulo: as 44640 entradas que foram, previamente, armazenadas na plataforma, em claro. Para efetuar este armazenamento prévio, foi construído um dispositivo produtor que enviou, diretamente, todos os dados obtidos do repositório

da Universidade da Califórnia Irvine. De seguida procedeu-se à recuperação desses dados. As medições correspondem ao tempo decorrido entre o pedido de todos os dados efetuado à plataforma e a receção desses dados no consumidor.

Recorrendo ao exemplo referido na secção anterior (Secção 6.1) e utilizando o código de processamento de dados apresentado na Secção 6.3.1, foram efetuadas medições de desempenho utilizando a arquitetura para uma plataforma IoT segura proposta nesta dissertação. De notar que apenas foram efetuadas medições de desempenho de acesso aos dados requisitado pelo consumidor. Só o desempenho no acesso aos dados é que faz sentido ser analisado uma vez que são as únicas medidas possíveis de serem comparadas com a utilização da arquitetura para uma plataforma IoT convencional. Todas as medições foram feitas através do cálculo da diferença existente entre tempos antes das operações iniciarem e depois das operações terminarem ao nível do código. Para as medições *end-to-end* foi efetuado o cálculo da diferença existente entre tempos antes do pedido do consumidor e depois da obtenção dos resultados. Foram efetuados vários tipos de medições. Em cada pedido do consumidor, foram medidos: tempos de pesquisa - os valores representam o tempo que foi necessário para que o componente de armazenamento da plataforma devolvesse os resultados contendo os dados cifrados pretendidos; tempos de transmissão para processamento - os valores representam o tempo que foi necessário para que os dados cifrados recuperados das bases de dados da plataforma fluam até à TTP de processamento para serem decifrados e processados; tempos de decifra - os valores representam o tempo que foi necessário para que a decifra dos dados devolvidos pela plataforma se efetuasse com sucesso; tempos de processamento - os valores representam o tempo que foi necessário para que o processamento dos dados utilizando o programa referido na Secção 6.3.1 concluísse e fossem recolhidos os resultados finais; tempos totais - os valores representam o tempo que foi necessário para que todo este processo de pedido de acesso aos dados por parte do consumidor concluísse. Relativamente aos tempos totais, é iniciado o cronómetro antes do consumidor efetuar o pedido e parado depois de receber os resultados.

Como foi possível verificar aquando da descrição da arquitetura proposta, o utilizador necessita de autorizar, expressamente, que este acesso aos seus dados se realize. Para efeitos de medição dos tempos, todas estas comunicações continuam a existir, no entanto a autorização expressa do utilizador não existe, simulando uma autorização efetuada de forma instantânea.

Para concretizar estas medições, todos os componentes são executados numa mesma máquina. O objetivo para tal decisão está no facto de diminuir ao máximo os atrasos provocados pela rede. Desta forma, é possível concentrar a análise dos tempos obtidos no processo de execução das várias tarefas pelos vários componentes não tendo interferência externa que pode ser muito variável. Contudo, os componentes, mesmo estando a ser

Tabela 6.2: Apresentação de medidas de desempenho medidos no acesso aos dados utilizando a arquitetura proposta

Medidas de Desempenho - Arquitetura Segura						**Média Aritmética (em segundos)**
	1	**2**	**3**	**4**	**5**	
Tempos de Pesquisa (em segundos)	13,569	16,346	16,176	16,708	16,931	15,946
Tempos de Transmissão para Processamento (em segundos)	7,840	8,330	8,22	6,560	7,580	7,706
Tempos de Decifra (em segundos)	3,676	5,672	4,516	4,158	3,788	4,362
Tempos de Processamento (em segundos)	0,061	0,137	0,091	0,065	0,071	0,085
Tempos Totais (em segundos)	30,817	36,418	36,041	33,841	34,645	34,352

executados na mesma máquina, comunicam normalmente através de HTTP e estão aptos para serem instalados e funcionar em máquinas diferentes.

Como é possível verificar na Tabela 6.2, que apresenta medidas de desempenho quando é utilizada a arquitetura segura proposta nesta dissertação, o tempo médio total de acesso aos dados é ligeiramente inferior ao dobro do tempo conseguido quando é utilizada uma plataforma IoT convencional. De forma a explicar tal valor, é útil analisar primeiro as medições de desempenho efetuadas às várias operações a nível individual.

No que toca ao tempo de processamento dos dados cifrados, este pode variar muito dependendo do algoritmo utilizado no processamento dos dados e no tipo e tamanho médio de resultados que se pretenda obter da execução deste algoritmo certificado em dados em claro. No exemplo aqui considerado, a execução do algoritmo sobre os dados não teve, praticamente, nenhum impacto no tempo total de acesso aos dados obtido, pelo que é irrelevante a sua análise. O algoritmo utilizado nesta simulação está presente na Figura 6.4. Os dados que estão a ser considerados são as 44640 entradas de medidas de consumo elétrico provenientes do repositório de dados mencionado anteriormente e totalizam 2,74 *megabytes*, em claro (sem quaisquer meta-dados associados). Estes foram os dados processados pelo código (certificado) exemplo mencionado na Figura 6.4.

Relativamente ao tempo de decifra, este já começa a ter algum impacto, mas não em demasiado, no tempo total despendido na operação de acesso aos dados de um utilizador. Cerca de 4,4 segundos, para a quantidade de dados que foram decifrados, tendo em conta o algoritmo utilizado e não tomando partido de aceleração de *hardware*, o tempo obtido não é mau e não contribui de forma significativa para o atraso das operações, pois apenas representa uma pequena fração (cerca de 12,7%) daquilo que é o tempo total necessário para efetuar o acesso aos dados.

Relativamente ao tempo médio de pesquisa dos dados, verifica-se que este valor é

muito semelhante ao tempo de pesquisa verificado na Tabela 6.1. Este facto era esperado uma vez que a operação de recuperação dos dados do componente de armazenamento persistente é necessária em ambas as arquiteturas. Esta operação de pesquisa demora cerca de 16 segundos. Isto é um pouco menos de metade do tempo total (que é de 34 segundos, em média, e com um desvio padrão de 2.234), cerca de 47%.

A principal razão para a existência de um tempo total tão alto tem que ver com a quantidade de dados que se está a tentar aceder, não por causa do processamento que é necessário ser efetuado sobre eles nem por causa da decifra que tem que ser, previamente executada, mas sim por causa da necessidade de transferência de todos estes dados cifrados e outros meta-dados de componentes para componentes. O tamanho dos dados total a ser recuperado e a ser transmitido do componente de armazenamento para a plataforma é de cerca de 29,3 *megabytes*. Este valor é alto uma vez que os dados obtidos do componente de armazenamento são os dados na sua forma cifrada, os meta-dados necessários devido à aplicação de mecanismos de cifra (instante de tempo de cifra e IV) e a presença de todos os meta-dados que uma plataforma IoT convencional necessita de adicionar para que a recuperação de dados se possa efetuar. A estrutura utilizada na representação destes dados é JSON. Como consequência deste aumento, o resultado é um atraso de cerca de 16 segundos (incluindo, obviamente os atrasos que são necessários na obtenção dos dados da própria base de dados do componente de armazenamento). Ora, na arquitetura que é aqui proposta, esta quantidade de dados tem de ser transferida, não uma vez, mas sim duas (Figura 5.11). Na solução proposta, os dados necessitam de ser transferidos através de HTTP desde o componente de armazenamento até à plataforma e da plataforma até à TTP de processamento utilizada. O tempo de transferência de todos estes dados cifrados foi medido e é de, aproximadamente, 7,7 segundos (incluindo conversões e processamento da estrutura de dados JSON em dicionários *python* para facilitar as restantes operações sobre os dados). Este tempo começa a ser significativo no impacto total (22% aproximadamente). De notar, que a comunicação existente na transferência de dados do componente de armazenamento para a plataforma IoT, não é local sendo que, na simulação efetuada, o componente de armazenamento não se encontra na mesma máquina, pelo que possíveis atrasos de rede adicionais poderão ter sido adicionados. Já na comunicação existente na transferência de dados da plataforma para a TTP de armazenamento, esta é feita internamente utilizando o mesmo sistema computacional diminuindo, ao máximo, possíveis atrasos de rede adicionais.

Considerando os tempos médios medidos, as operações, cujas medidas foram apresentadas, representam cerca de 82% do tempo total. O tempo restante, para perfazer o tempo total de acesso aos dados, corresponde a todas as outras trocas de informação e seu processamento como pedidos de chaves, criação e envio de asserções SAML, entre outros.

De notar também que, como observado, as medidas variam muito de execução para execução (desvio padrão de cerca de 2.23) dependendo das condições de rede no momento e outros fatores, pelo que não é possível deduzir com a maior precisão, os tempos de desempenho que é necessário no acesso aos dados de um utilizador.

Concluindo esta secção, é possível verificar que era difícil obter valores de desempenho menores àqueles obtidos na Tabela 6.1, uma vez que a aplicação de mecanismos de segurança afeta sempre este tipo de requisitos importantes e ao facto das operações executadas na Tabela 6.1 estarem incluídas nas operações executadas utilizando uma arquitetura segura. No entanto, os valores totais obtidos não são maus. Principalmente, pelo facto de que estes acessos serão, na sua maior parte, efetuados com o objetivo de fazer estudos de consumo de utilizadores (no caso de uso que se tem vindo a considerar) e outras operações onde a rapidez de acesso aos resultados não é uma prioridade.

6.3 USABILIDADE

Nesta secção serão descritas situações de usabilidade da arquitetura proposta nesta dissertação. Existem duas perspetivas a ter em conta nesta questão. Existe a perspetiva de usabilidade da plataforma IoT do consumidor/fornecedor de serviços ou do utilizador/cliente respetivo do seus fornecedor.

6.3.1 CONSUMIDOR/FORNECEDOR DE SERVIÇOS

A entidade consumidor de dados ou fornecedor de serviços, no caso de se estar a considerar o caso de uso de medidores de energia elétrica inteligentes, como referido no início deste capítulo, têm de ser capazes de, facilmente, poderem utilizar a plataforma IoT implementada segundo a arquitetura proposta.

Primeiramente, para o consumidor ser capaz de utilizar a arquitetura segura de forma a poder obter dados transformados dos seus clientes (utilizando-os para, por exemplo, proceder a diversos estudos de consumo), tem que fabricar o respetivo código que irá ser utilizado para efetuar a transformação dos dados. Um exemplo de código relevante (neste caso escrito na linguagem *Python*, no entanto a linguagem *Java* é também suportada) é mostrado de seguida:

Este é um exemplo de código relevante que irá ser executado na TTP de processamento.

Uma vez que a TTP de certificação de código irá analisar manualmente e certificar que este código é correto e não põe a causa a privacidade dos utilizadores, controlando

```
timesDict = {}
for aResult in theResults:
  splitted = aResult['data'].split(';', 3)
  timeOfDay = splitted[1]
  globalAPower = splitted[2]
  if timeOfDay not in timesDict:
    timesDict[timeOfDay] = float(globalAPower)
  else:
    timesDict[timeOfDay] += float(globalAPower)
processedResults = {'popularTime' : max(timesDict, key=timesDict.get)}
```

Figura 6.4: Código exemplo utilizado no processamento de dados em claro

a forma como os seus dados são transformados, o programa em causa necessita de ser simples e não constar de diversos módulos e de confusão adicional para os seres humanos que irão efetuar a rigorosa análise e verificação. Isto é importante, dado que diminui, em muito, a probabilidade de ser emitida uma certificação de código a um programa malicioso erradamente, pois torna-se mais clara a sua análise.

Este pequeno programa processa os dados em claro que recebe. Antes deste processamento, foi levado a cabo um processo de validação de certificados, asserções SAML, pedido de chaves à TTP de autorização para a decifra dos dados que recebe da plataforma e, finalmente, decifra estes mesmos dados utilizando o algoritmo apropriado (AES em modo CFB de momento). Como já foi mencionado nesta dissertação, neste momento o código a ser enviado pelo consumidor para a TTP de processamento terá de ter todas estas funções. Isto faz com que esta TTP seja muito simples onde o seu único propósito é a criação do ambiente de execução e a respetiva execução do programa. Este facto permite, ainda, que o consumidor possa escolher os algoritmos de decifra a serem utilizados. No entanto, como trabalho futuro, poderá ser possível agilizar estes procedimentos incluindo estas funções numa API instalada de modo nativo.

De um modo geral, este código irá processar informação semelhante aos dados originais enviados pelos dispositivos produtores/*gateways* semelhantes ao exemplo de dados produzidos mencionado na Secção 6.1. Este programa irá analisar o consumo elétrico de utilizadores. Este consumo é medido de minuto a minuto. Num determinado dia, existirão assim instantes de tempo em que a residência do utilizador gastou mais energia do que outros.

Este programa irá analisar um conjunto de dias delimitado por dois instantes de tempo definidos pelo consumidor no momento do pedido (não no momento de criação deste programa). No final, irá ser calculado o momento do dia (instante de tempo em horas e minutos) em que o cliente do fornecedor de energia mais gasta eletricidade. Este

cálculo é feito analisando os dados de vários dias e encontrando o máximo de energia gasta para um determinado instante de tempo. Desta forma, o programa começa por iterar sobre todos os dados decifrados (`theResults`). Como cada unidade de dados separa a sua informação utilizando o ponto e vírgula (`;`), é efetuada um divisão obtendo o instante de tempo em que os dados foram recolhidos e os próprios dados. Após efetuada esta divisão, analisa-se o consumo de energia elétrica naquele determinado momento adicionando uma entrada a um dicionário (*array* associativo), no início criado, em que a chave é o instante de tempo e o valor é o consumo de energia. No caso do instante de tempo já existir, são acumulados os valores de energia consumidos. No final é obtido o instante de tempo em que o valor de energia elétrica consumido é máximo.

Para este simples exemplo, o processamento está concluído. Estes resultados, após serem cifrados, fluirão para o consumidor, pelo que só o consumidor tem acesso aos dados processados. Desta forma, o fornecedor de energia elétrica nunca obterá detalhes suficiente, dependendo obviamente do processamento efetuado e certificado pela entidade de certificação externa, para que se possa pôr em causa a segurança e a privacidade dos seus clientes. Ao mesmo tempo estes dados resultantes são úteis para efetuar certos estudos e obter informação estatística.

6.3.2 UTILIZADOR/CLIENTE

O utilizador acede, através de certas interfaces gráficas com vários componentes. No entanto, o seu acesso à interface do consumidor é extremamente variável. De entre todas as operações que o utilizador pode efetuar no consumidor, uma delas passa pelo registo de novo *hardware*. Este registo é simples pelo que apenas é necessário informar o consumidor de que determinadas TTPs existem (e quais os seus endereços e portas de acesso) e efetuar uma associação entre TTP de autorização e *gateways* que tenha em sua posse. Para além deste registo e associação, o utilizador deve ser capaz de poder aceder a informação sobre um determinado programa certificado que irá processar os seus dados. Este programa é identificado recorrendo a uma impressão digital (*digest*). Na tentativa de obtenção da autorização do utilizador para aceder aos seus dados e processá-los utilizando um determinado código certificado, o utilizador é apresentado com uma impressão digital do código que deverá estar disponível para consulta opcional no respetivo fornecedor de serviços do cliente em causa.

Para além disso, o *gateway* (ou um dispositivo produtor que tenha as mesmas capacidades de um *gateway* e tenha em si implementados os mecanismos necessários para que seja possível fazer uso da arquitetura proposta) necessitará de um meio para expor 4 caracteres variáveis ao utilizador. Estes serão necessários na fase de registo

(inicialização do *gateway*). Tal pode ser comunicado ao utilizador através de um pequeno ecrã OLED. Tudo o que o utilizador necessita de fazer é retirar a informação pretendida simplesmente olhando para o dispositivo *gateway* que, fisicamente, estará presente em um local seguro e privado para o utilizador, tipicamente a sua residência.

Adicionalmente, o utilizador necessitará de aceder à interface do componente de autorização. É vital que este permita que o utilizador autorize ou não registos de *gateways* (seu equipamento). O utilizador, através deste componente, pode tomar este tipo de decisões. No caso do registo do *gateway* o utilizador terá de verificar na interface da TTP que os 4 caracteres apresentados são os mesmos que os apresentados no dispositivo *gateway* e aceitar tal registo ou não. Desta forma, é possível garantir que os dois componentes estão realmente a comunicar um com o outro e nenhum deles é um impostor. Para além disso, permite ainda assegurar que não é possível executar o ataque *man-in-the-middle*. Adicionalmente, sempre que um consumidor pretenda efetuar o acesso aos dados de um utilizador, o último necessitará de, presencialmente, autorizar este acesso indicando apenas uma resposta afirmativa ou negativa para tal. Por último existem diversas operações, não especificadas detalhadamente nesta dissertação, que o utilizador pode efetuar, opcionalmente, nesta TTP. Estas operações podem envolver a manipulação das chaves simétricas armazenadas. O utilizador poderá, por exemplo fazer a transferência de todas as chaves para que as possa ter em sua posse (*backup*), ou pode, por exemplo eliminar um conjunto definido de chaves não permitindo, nunca mais, que os dados produzidos cifrados com essas chaves, sejam acessíveis novamente. Um dos objetivos nesta dissertação era o de fornecer ao utilizador mecanismos úteis de segurança para que este pudesse ter controlo sobre os seus dados que produz. Com esta arquitetura e com as vantagens na usabilidade na perspetiva do utilizador, tal foi conseguido.

6.4 DISTRIBUIÇÃO DE PODER

Como foi referido na Secção 4.1.7, um dos problemas que existe na maioria das arquiteturas que contemplam a segurança e em que se lida com várias entidades diferentes, é o problema da distribuição dos tipos de dados existentes no sistema por cada uma dessas entidades. Tal distribuição tem de ser feita de forma a que nenhuma entidade tenha o poder de quebrar a segurança dos dados no sistema. Ou seja, nenhuma entidade pode ter acesso a toda a informação necessária para que a mesma possa decifrar dados e pôr em causa a privacidade dos utilizadores donos dos mesmos.

Assim, na Tabela 6.3, nas colunas estão representadas as diferentes entidades existentes na arquitetura que é proposta nesta dissertação. Nas linhas estão representados os

Tabela 6.3: Indicação de a que tipo de dados é que, cada componente relevante na arquitetura proposta, tem acesso

Componentes	Consumidor	Plataforma IoT	TTP de Autorização	TTP de Processamento	Dispositivos / Gateways
Utilizadores	**X**	**(X)**	**X**		
Chaves Secretas			**X**	**(X)**	**(X)**
Dados Cifrados		**X**		**X**	**X**
Dados em Claro				**X**	**X**
Dados Processados	**X**			**X**	

diferentes tipos de informação existentes na arquitetura segura. Nas diferentes células, caso a respetiva entidade tenha em sua posse o respetivo tipo de informação, tal é representado com um **X**. Caso contrário a célula da tabela é vazia. No caso em que uma entidade possua ou tenha acesso a parte ou temporariamente um determinado tipo de informação, tal é indicado na respetiva célula com um **(X)**.

Na tabela apresentada, verificamos que nenhuma entidade tem acesso a tipos de informação suficientes para que possa violar a privacidade de um utilizador ao aceder à informação.

O consumidor tem de ter obrigatoriamente em sua posse a informação dos diferentes utilizadores. Estes utilizadores são seus clientes e é deles que o consumidor (o fornecedor de serviços) irá obter a transformação dos dados, que preserva a privacidade, que deseja. Para além da informação dos utilizadores, o consumidor irá também ter acesso aos dados processados pela TTP de processamento utilizando um programa certificado. O consumidor, apenas com o acesso a este tipo de informação nunca poderá pôr em causa a segurança dos dados e a privacidade dos utilizadores. Os dados processados são suficientemente anónimos para que o consumidor não possa retirar conclusões perigosas sobre os seus clientes. Estes dados processados, como foi visto, são o resultado do processamento do código certificado. O consumidor não tem acesso aos dados cifrados, no entanto, não é difícil planejar um ataque para que se obtenha este tipo de informação. Assim, implicitamente e na elaboração da arquitetura, foi assumido que todas as entidades podem, eventualmente, ter acesso aos dados cifrados. Para além disso, o consumidor não tem acesso a nenhumas chaves secretas usadas nas cifras

dos dados que são efetuadas pelos dispositivos/*gateways* e que são armazenadas na plataforma, pelo que não poderá, mesmo tendo acesso aos dados originais cifrados, decifrar e obter informação sensível. Isto leva, evidentemente, ao principal tipo de informação a que nenhuma entidade (excetuando os componentes do domínio seguro do utilizador) pode ter acesso: os dados em claro.

A plataforma IoT tem acesso a informação do utilizador que é passada, apenas, numa primeira fase de registo. Esta informação consiste apenas no seu identificador. Na arquitetura que é proposta, a plataforma não teria acesso a quaisquer tipos de dados do utilizador pelo que não conseguiria obter informações sobre o mesmo. No entanto, mesmo que a plataforma obtenha, eventualmente, esta informação (algo que pode acontecer em plataformas IoT não seguras atualmente existentes), ela nunca poderá obter informação detalhada acerca de um determinado utilizador e associando-o a dados produzidos, pois os últimos passam e são armazenados sempre pela plataforma na sua forma cifrada. Para além disso, a plataforma não tem acesso a quaisquer chaves secretas, pois estas fluem cifradas pela plataforma. Como tal, os dados originais em claro nunca estarão disponíveis para a plataforma. Por último, nem mesmo os dados processados estão acessíveis à plataforma. Embora possa parecer que estes dados não causam, à partida, problemas de segurança, caso sejam expostos à plataforma, a última não necessita de lhes aceder para que funcione corretamente e de acordo com o esperado. Este é o princípio *Need-to-Know* que nos diz que as diversas entidades apenas devem ter acesso à informação que precisam de forma a conseguirem executar corretamente as suas funções e processamento contribuindo para o bom funcionamento de todo o sistema.

A TTP de autorização irá necessitar de ter acesso à informação do utilizador. Uma vez que se trata de uma entidade externa confiável, o utilizador necessita de estar registado nessa entidade. Esta TTP tem que ter em si armazenado um perfil e um armazenamento de dados para cada utilizador, fazendo esta separação, ao contrário da TTP de processamento. Desta forma, terá acesso a informação do cliente incluindo a sua identificação. A esta identificação estarão associados determinados identificadores de *gateways* ou outros dispositivos encarregados de efetuar a cifra dos dados. Mais uma vez, toda esta informação estará associada a chaves criptográficas simétricas secretas. Se a TTP de autorização tiver acesso aos dados cifrados (o que se supõe que todas as entidades o podem ter por meios de ataques a eventuais meios de comunicação utilizados na troca de dados entre os dispositivos e a plataforma, por exemplo), com a ajuda das chaves secretas e da identidade dos utilizadores, a TTP poderá pôr em causa a privacidade dos utilizadores da plataforma. Desta forma, é imperativo que esta entidade aja corretamente e que seja confiável, caso seja utilizado um serviço externo para fins de autorização. A TTP de autorização não possui acesso aos dados em claro,

se não tiver em sua posse os dados cifrados de um utilizador. Finalmente, esta entidade está encarregue de fazer interface com o utilizador de forma a autorizar acesso aos dados do mesmo, pelo que, não estará ligada diretamente com o processamento (generalização e anonimato) dos dados e, por isso, não tem acesso a quaisquer dados processados.

A TTP de processamento não irá necessitar de ter acesso à informação dos utilizadores. No entanto, acederá aos dados cifrados e às chaves secretas que os decifram. Desta forma, poderá, obviamente, decifrar os dados e ver os dados originais na sua forma clara. Este componente não separa os utilizadores no seu domínio, assegurando apenas que os dados que recebe, juntamente com a informação da TTP de autorização para pedidos de chaves, são processados nas suas máquinas. Os dados estarão à mercê do código certificado, o que produzirá, como resultado, dados processados. Como foi visto, este componente tem acesso a todos estes dados à exceção de informação do utilizador. Logo, no pior dos casos, esta entidade poderá obter uma coleção de dados sem que estes estejam associados a nenhum utilizador ou cliente de um fornecedor de serviços (no caso de uso considerado). À partida, tal poderá não representar um problema de segurança no sistema. No entanto, de forma a maximizar a segurança e evitar vulnerabilidades ou uma eventual fuga de informação, caso o componente de processamento não seja privado e não pertença ao domínio de segurança (residencial) do utilizador, este deve pertencer a uma entidade externa confiável (TTP).

Os dados que os dispositivos/*gateways* possuem não são relevantes para a garantia de segurança dos dados dos utilizadores uma vez que estes componentes pertencem ao próprio utilizador e estão, tipicamente localizados na sua residência, uma zona privada e segura. No entanto, vale a pena referir que os tipos de informação a que estes dispositivos têm acesso são aos dados cifrados (uma vez que são estes dispositivos que os cifram), aos dados em claro (uma vez que são estes dispositivos que produzem os dados) e a uma determinada chave secreta (aquela que está a utilizar para cifrar os seus dados). Caso os componentes de autorização e de processamento não pertençam a uma entidade externa confiável (TTP), a análise de distribuição de poderes seria também irrelevante para estes componentes uma vez que todos pertenceriam ao utilizador e estariam dentro de uma zona segura.

Tendo em conta todos estes aspetos, é difícil, para um atacante (pertencente a uma das entidades ou não), obter informação relevante dos utilizadores da plataforma IoT. Tal só poderia acontecer no caso da existência de conluio entre as entidades. Ou seja, no caso de existir cooperação e partilha ilegal de dados provenientes, pelo menos, de duas entidades diferentes na arquitetura segura proposta.

capítulo 7

Conclusão

Neste capítulo, são retiradas algumas conclusões resultantes do trabalho executado nesta dissertação.

Após ter sido executado este trabalho e, em jeito de conclusão, é importante retirar algumas conclusões.

Uma arquitetura segura não é fácil de ser elaborada. Existem vários pontos de falha e, na maioria das vezes, é impossível cobrir todos os ataques e vulnerabilidades possíveis. Tendo isto em consideração, o facto de não estar a ser elaborada uma arquitetura a partir do zero, acrescenta ainda mais dificuldade na elaboração de uma arquitetura deste género. As arquiteturas existentes atualmente preocupam-se e focam-se muito mais no problema da funcionalidade do que no problema da segurança do sistema, principalmente ao nível dos dados e, como se sabe, a segurança é inimiga da funcionalidade, desempenho, sobrecarga e usabilidade. É muito difícil, para não dizer praticamente impossível, de se conseguir estabelecer uma arquitetura que implemente mecanismos de segurança e que não produza impacto negativo nestes aspetos que as plataformas IoT devem proporcionar. O desempenho é sempre afetado uma vez que estamos a lidar com cifras, decifras e trocas de chaves. A sobrecarga acontece sempre pois existe informação associada à cifra de dados que tem de ser armazenada juntamente com os mesmos, nomeadamente um IV e a adição de controlo de integridade desses dados. A usabilidade é afetada uma vez que são adicionadas novas entidades o que aumenta a complexidade do sistema e o número de operações de registo que é necessário executar pelos intervenientes humanos. Finalmente, todas estes aspetos contribuem para uma diminuição generalizada da funcionalidade do sistema.

Desta forma, e sabendo que as arquiteturas originais foram construídas a pensar na funcionalidade acima de tudo, torna-se ainda mais difícil conseguir estabelecer uma

arquitetura que permita adicionar segurança ao nível dos dados e da privacidade do utilizador, principalmente. Para além disso, existiram ainda outras restrições que tiveram de ser analisadas, como o problema da capacidade de comunicação entre o domínio do utilizador (dispositivos e *gateways* são fabricados de modo a utilizarem redes móveis através de um cartão SIM).

Foram adicionadas diversas entidades novas neste tipo de arquitetura. Idealmente, os componentes de autorização e de processamento deveriam pertencer ao domínio de segurança do utilizador, no entanto, foi considerado que nem todos os utilizadores têm a capacidade económica de ter esse tipo de dispositivos em sua posse ou na sua residência e, considerou-se que tais serviços seriam prestados por entidades externas confiáveis. A existência de confiança por entidades terceiras nunca é desejável, pois não pode ser validada. No entanto, não foi possível encontrar uma melhor solução segura que não envolvesse estas entidades, no caso em que o próprio utilizador, na maioria das vezes e por razões económicas, não pode ter na sua residência um novo dispositivo que trate de efetuar as operações que o componente de processamento e de autorização fazem.

Revelou-se que, apesar de todo o impacto negativo verificado, foi possível obter uma arquitetura de segurança no ambiente da IoT que resolve vários problemas de segurança dos dados produzidos e controlo de acesso aos mesmos tornando possível que o utilizador dono dos seus dados, possa controlá-los e nunca perder esse controlo sobre os mesmos sabendo sempre quais dados proprietários é que estão a ser acedidos e processados utilizando programas certificados. Para além disso nenhuma entidade pode ter acesso aos seus dados originais, mas apenas a uma transformação, generalização e anonimato dos mesmos produzido pelo código certificado. Certificação feita por uma entidade terceira confiável que trata de analisar o programa, manualmente e recorrendo a recursos humanos, para que seja garantida a privacidade dos utilizadores. Como medida adicional de controlo sobre os dados, os utilizadores podem ainda autorizar ou não o acesso à sua produção, facilmente, através da interação com o componente de autorização associado.

Relativamente à implementação efetuada, foi desenvolvido um protótipo que mostra todas as operações e interações entre os diversos componentes considerados na Secção 5.4. Foi desenvolvido um consumidor de dados que efetua operações de registo na plataforma, na TTP de autorização e na TTP de processamento. Para além disso, pode registar código certificado na plataforma. Por último, a principal função de aceder aos dados de dispositivos produtores de um utilizador foi também implementada. Foi também desenvolvido um dispositivo produtor. Este trata de de obter informação sobre os parques de estacionamento da Universidade de Aveiro recorrendo aos *webservices* da universidade que são disponibilizados. Estes dados são enviados de 5 em 5 segundos para um *gateway*, também criado de raíz. Este *gateway* tem funções de receção de

dados de dispositivos e funções de registo na TTP de autorização associada. Para além disso, o *gateway* tem funções de pedido de chaves à TTP de autorização, cifra de dados e envio dos dados cifrados para a plataforma IoT. Foi também implementado de raíz o Mediador que trata de obter a autorização do utilizador (comunicando com a TTP de autorização), trata também de recuperar os dados do componente de armazenamento já existente, enviá-los para a respetiva TTP de processamento e recolher os resultados da transformação dos dados. Foi também criada de raíz a TTP de processamento que trata de executar código certificado e proceder à transformação dos dados em claro pertencentes ao utilizador (dono dos dados) num resultado sem detalhe suficiente que possa pôr em causa a privacidade do utilizador. A TTP de autorização foi também criada de raíz com funcionalidades de criação, armazenamento e disponibilização de chaves autorizadas previamente pelo utilizador. Por último foram criadas APIs da plataforma assim como o componente de registo mencionado na Secção 5.4.7. Todas as interações ente todos os componentes apresentadas na Secção 5.5 foram implementadas, com a exceção de interações com o utilizador (interfaces com o utilizador) e operações de registo de utilizadores nos serviços de terceiros que, teoricamente, terão de existir num sistema real. Foi, então, feita uma simulação da arquitetura proposta utilizando um conjunto de dados de consumo elétrico que foi enviado e armazenado na plataforma na sua forma cifrada fazendo uso dos mecanismos implementados. De seguida todos esses dados foram requeridos por um consumidor que utilizou o código certificado apresentado na Figura 6.4. Após todas as passagens de mensagens entre os diversos componentes e da autorização do utilizador dono dos dados, estes dados foram processados pelo código e foi obtido um resultado que foi retornado ao consumidor. O resultado da execução deste código, mostrou que, durante o mês de janeiro de 2007, a hora do dia em que a residência consumiu mais energia elétrica foi às 20:54. Esta informação foi a única informação obtida pelo fornecedor de serviços do seu cliente. Desta forma, garante-se a privacidade do utilizador enquanto que o consumidor pode obter informação útil.

Existem alguns aspetos na arquitetura proposta que poderiam ser melhorados. Estes foram mencionados anteriormente e incluídos em trabalho futuro. Melhorias como a implementação de uma API nativamente instalada nos TTPs de processamento. Desta forma, o código certificado teria apenas acesso a funções dessa API para obtenção de chaves, decifra, envio de resultados para a rede e outros, sem ter acesso a possíveis outras funções do sistema computacional em que está a executar. Uma outra melhoria centra-se na emissão de uma asserção SAML por parte da entidade de certificação de código onde estabeleceria certas condições de utilização do código e estes termos e condições teriam de ser aceitados pelo utilizador para que o processo de obtenção de dados avançasse. Por último, o pedido de chaves que parte da TTP de processamento para a TTP de autenticação, poderia ser incluído no momento do pedido de autorização

feito pela plataforma à entidade de autorização. Tal resultaria em um pedido e resposta a menos o que poderia aumentar o desempenho.

Nesta dissertação construiu-se uma arquitetura para plataformas IoT segura. Para tal, foi tido sempre em mente um princípio de segurança neste tipo de sistemas: a distribuição de poder. Conseguindo efetuar uma distribuição de informação útil por diversas entidades (mesmo que externas), é possível aumentar a segurança global do sistema, pois nenhuma entidade interveniente pode, por si só, por em causa a privacidade dos utilizadores.

Referências

[1] 1ª ed. Lopez Research LLC, 2013, acedido em julho de 2016. endereço: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/introduction_to_IoT_november.pdf.

[2] 1ª ed. ETSI, 2013, acedido em julho de 2016. endereço: https://www.etsi.org/deliver/etsi_ts/102600_102699/102690/02.01.01_60/ts_102690v020101p.pdf.

[3] 1ª ed. Cisco, 2014, acedido em julho de 2016. endereço: http://cdn.iotwf.com/resources/71/IoT_Reference_Model_White_Paper_June_4_2014.pdf.

[4] 1ª ed. Wind River, 2015, acedido em julho de 2016. endereço: http://www.windriver.com/whitepapers/security-in-the-internet-of-things/wr_security-in-the-internet-of-things.pdf.

[5] D. E. 3rd e T. Hansen, *Us secure hash algorithms (sha and sha-based hmac and hkdf)*, RFC 6234 (Informational), Internet Engineering Task Force, mai. de 2011. endereço: http://www.ietf.org/rfc/rfc6234.txt.

[6] *50 sensor applications for a smarter world*, http://www.libelium.com/top_50_iot_sensor_applications_ranking/, acedido em julho de 2016, 2016.

[7] R. Arends, R. Austein, M. Larson, D. Massey e S. Rose, «Dns security introduction and requirements», rel. téc., 2005.

[8] Assembleia da República Portuguesa, «Lei n.o 67/98 de 26 de outubro, lei da protecção de dados pessoais», *Diário da República, Nº 247, Série I-A*, 1998, https://www.cnpd.pt/bin/legis/nacional/LPD.pdf.

[9] J. Bethencourt, A. Sahai e B. Waters, «Ciphertext-policy attribute-based encryption», em *2007 IEEE symposium on security and privacy (SP'07)*, IEEE, 2007, pp. 321–334.

[10] T. Bray, *The javascript object notation (json) data interchange format*, RFC 7159 (Proposed Standard), Internet Engineering Task Force, mar. de 2014. endereço: http://www.ietf.org/rfc/rfc7159.txt.

[11] J. Cao, B. Carminati, E. Ferrari e K.-L. Tan, «Castle: Continuously anonymizing data streams», *IEEE Transactions on Dependable and Secure Computing*, vol. 8, nº 3, pp. 337–352, 2011. DOI: 10.1109/tdsc.2009.47.

[12] B. Chor, E. Kushilevitz, O. Goldreich e M. Sudan, «Private information retrieval», *Journal of the ACM (JACM)*, vol. 45, nº 6, pp. 965–981, 1998.

[13] Council of European Union, «Regulation (EU) 2016/679 of the european parliament and of the council», 2016, http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG.

[14] B. P. Crow, I. Widjaja, L. Kim e P. T. Sakai, «Ieee 802.11 wireless local area networks», *IEEE Communications magazine*, vol. 35, nº 9, pp. 116–126, 1997.

[15] J. Daemen e V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[16] «Dcapbac: Embedding authorization logic into smart things through ecc optimizations», *International Journal of Computer Mathematics*, vol. 93, nº 2, pp. 345–366, 2016.

[17] C. Demerjian, *Intel lets you manipulate encrypted data*, acedido em julho de 2016, 2012. endereço: http://semiaccurate.com/2012/06/27/intel-lets-you-manipulate-encrypted-data/.

[18] R. H. Deng, *Flexible access of encrypted data in the cloud*, acedido em julho de 2016, 2013. endereço: http://www.ieccr.net/2013/pairing2013/Robert-deng.pdf.

[19] T. Dierks, «The transport layer security (tls) protocol version 1.2», 2008.

[20] J. Dixon, *Who will step up to secure the internet of things?*, acedido em julho de 2016, 2015. endereço: http://techcrunch.com/2015/10/02/who-will-step-up-to-secure-the-internet-of-things/.

[21] C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou e G. Vassilacopoulos, «Enabling data protection through pki encryption in iot m-health devices», *2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE)*, 2012. DOI: 10.1109/bibe.2012.6399701.

[22] U. Feige, A. Fiat e A. Shamir, «Zero-knowledge proofs of identity», *Journal of cryptology*, vol. 1, nº 2, pp. 77–94, 1988.

[23] D. Ferraiolo, J. Cugini e D. R. Kuhn, «Role-based access control (rbac): Features and motivations», em *Proceedings of 11th annual computer security application conference*, 1995, pp. 241–48.

[24] L. Ferretti, M. Colajanni, M. Marchetti, A. E. Scaruffi e D. SpA, «Transparent access on encrypted data distributed over multiple cloud infrastructures», *The Fourth International Conference on Cloud Computing, GRIDs, and Virtualization, CLOUD COMPUTING*, pp. 201–207, 2013.

[25] C. Gentry, «Computing arbitrary functions of encrypted data», *Communications of the ACM*, vol. 53, nº 3, p. 97, 2010. DOI: 10.1145/1666420.1666444.

[26] V. Goyal, O. Pandey, A. Sahai e B. Waters, «Attribute-based encryption for fine-grained access control of encrypted data», em *Proceedings of the 13th ACM conference on Computer and communications security*, Acm, 2006, pp. 89–98.

[27] A. Greenberg, *An mit magic trick: Computing on encrypted databases without ever decrypting them*, acedido em julho de 2016, 2011. endereço: http://www.forbes.com/sites/andygreenberg/2011/12/19/an-mit-magic-trick-computing-on-encrypted-databases-without-ever-decrypting-them/#3658fd141ce4.

[28] U. Greveler, P. Glösekötter, B. Justus e D. Loehr, «Multimedia content identification through smart meter power usage profiles», *Computers, Privacy and Data Protection CPDP*, 2012.

[29] J. C. Haartsen, «The bluetooth radio system», *IEEE personal communications*, vol. 7, nº 1, pp. 28–36, 2000.

[30] N. Hajdarbegovic, *Are we creating an insecure internet of things (iot)? security challenges and concerns*, acedido em julho de 2016, 2015. endereço: http://www.toptal.com/it/are-we-creating-an-insecure-internet-of-things.

[31] D. Hardt, «The oauth 2.0 authorization framework», 2012.

[32] M. Henze, L. Hermerschmidt, D. Kerpen, R. Haussling, B. Rumpe e K. Wehrle, «User-driven privacy enforcement for cloud-based services in the internet of things», *2014 International Conference on Future Internet of Things and Cloud*, 2014. DOI: 10.1109/ficloud.2014.38.

[33] M. Henze, R. Hummen e K. Wehrle, «The cloud needs cross-layer data handling annotations», *Security and Privacy Workshops (SPW), 2013 IEEE*, pp. 18–22, 2013.

[34] J. Hernández-Ramos, J. Bernabe, M. Moreno e A. Skarmeta, «Preserving smart objects privacy through anonymous and accountable access control for a m2m-enabled internet of things», *Sensors*, vol. 15, nº 7, pp. 15 611–15 639, 2015. DOI: 10.3390/s150715611.

[35] R. Housley, W. Polk, W. Ford e D. Solo, «Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile», rel. téc., 2002.

[36] X. Huang, R. Fu, B. Chen, T. Zhang e A. W. Roscoe, «User interactive internet of things privacy preserved access control», *Internet Technology And Secured Transactions, 2012 International Conference for Internet Technology and Secured Transactions*, pp. 597–602, 2012.

[37] W. S. Inbarani, G. Shenbagamoorthy e C. K. C. Paul, «Proxy re-encryption schemes for data storage security in cloud-a survey», *International Journal of Engineering Research and Technology*, vol. 2, nº 1, 2013.

[38] W. Itani, A. Kayssi e A. Chehab, «Privacy as a service: Privacy-aware data storage and processing in cloud computing architectures», *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2009. DOI: 10.1109/dasc.2009.139.

[39] M. E. Jobst, «Security and privacy in the smart energy grid», *Network*, vol. 159, 2013.

[40] V. Kapoor, V. S. Abraham e R. Singh, «Elliptic curve cryptography», *Ubiquity*, vol. 2008, nº May, p. 7, 2008.

[41] J. Kincaid, «Urban airship brings easy push notifications to android», *TechCrunch, Aug*, vol. 10, 2010.

[42] P. Kinney et al., «Zigbee technology: Wireless control that simply works», em *Communications design conference*, vol. 2, 2003, pp. 1–7.

[43] A. Kivity, Y. Kamay, D. Laor, U. Lublin e A. Liguori, «Kvm: The linux virtual machine monitor», em *Proceedings of the Linux symposium*, vol. 1, 2007, pp. 225–230.

[44] D. Kozlov, J. Veijalainen e Y. Ali, «Security and privacy threats in iot architectures», *Proceedings of the 7th International Conference on Body Area Networks*, pp. 256–262, 2012.

[45] H. Krawczyk, R. Canetti e M. Bellare, «Hmac: Keyed-hashing for message authentication», 1997.

[46] P. J. Leach, M. Mealling e R. Salz, «A universally unique identifier (uuid) urn namespace», 2005.

[47] J. Linn, *Privacy enhancement for internet electronic mail: part i: message encryption and authentication procedures*, RFC 1421 (Historic), Internet Engineering Task Force, fev. de 1993. endereço: http://www.ietf.org/rfc/rfc1421.txt.

[48] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet e D. Irwin, «Private memoirs of a smart meter», *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building - BuildSys '10*, 2010. DOI: 10.1145/1878431.1878446.

[49] B. C. Neuman e T. Ts'o, «Kerberos: An authentication service for computer networks», *IEEE Communications magazine*, vol. 32, nº 9, pp. 33–38, 1994.

[50] N. Ragouzis, J. Hughes, R. Philpott, E. Maler, P. Madsen e T. Scavo, «Security assertion markup language (saml) v2. 0 technical overview», *OASIS Comittee Draft*, vol. 2, p. 2008, 2008.

[51] E. Ramirez, *Opening remarks of ftc chairwoman edith ramirez - privacy and the iot: Navigation policy issues*, acedido em julho de 2016, 2015. endereço: https://www.ftc.gov/system/files/documents/public_statements/962063/160621aolarremarks.pdf.

[52] J. V. G. Rebelo e H. A. S. Gomes, «Data encryption standard»,

[53] M. G. Reed, P. F. Syverson e D. M. Goldschlag, «Anonymous connections and onion routing», *IEEE Journal on Selected areas in Communications*, vol. 16, nº 4, pp. 482–494, 1998.

[54] E. Rescorla e N. Modadugu, «Datagram transport layer security version 1.2», 2012.

[55] A. Rial e G. Danezis, «Privacy-preserving smart metering», *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society - WPES '11*, 2011. DOI: 10.1145/2046556.2046564.

[56] R. L. Rivest, A. Shamir e L. Adleman, «A method for obtaining digital signatures and public-key cryptosystems», *Communications of the ACM*, vol. 21, nº 2, pp. 120–126, 1978.

[57] R. Roman, J. Zhou e J. Lopez, «On the features and challenges of security and privacy in distributed internet of things», *Computer Networks*, vol. 57, nº 10, pp. 2266–2279, 2013. DOI: 10.1016/j.comnet.2012.12.018.

[58] A. Sahai, «Computing on encrypted data», *Information Systems Security*, pp. 148–153, 2008.

[59] B. K. Samanthula, G. Howser, Y. Elmehdwi e S. Madria, «An efficient and secure data sharing framework using homomorphic encryption in the cloud», em *Proceedings of the 1st International Workshop on Cloud Intelligence*, ACM, 2012, p. 8.

[60] L. Sankar, S. Rajagopalan, S. Mohajer e H. Poor, «Smart meter privacy: A theoretical framework», *IEEE Trans. Smart Grid*, vol. 4, nº 2, pp. 837–846, 2013. DOI: 10.1109/tsg.2012.2211046.

[61] Z. Shelby, K. Hartke e C. Bormann, *The constrained application protocol (coap)*, RFC 7252 (Proposed Standard), Internet Engineering Task Force, jun. de 2014. endereço: http://www.ietf.org/rfc/rfc7252.txt.

[62] S. Sicari, A. Rizzardi, L. Grieco e A. Coen-Porisini, «Security, privacy and trust in internet of things: The road ahead», *Computer Networks*, vol. 76, pp. 146–164, 2015. DOI: 10.1016/j.comnet.2014.11.008.

[63] S. Sicari, C. Cappiello, F. De Pellegrini, D. Miorandi e A. Coen-Porisini, «A security-and quality-aware system architecture for internet of things», *Information Systems Frontiers*, 2014. DOI: 10.1007/s10796-014-9538-x.

[64] L. Sweeney, «K-anonymity: A model for protecting privacy», *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, nº 05, pp. 557–570, 2002.

[65] G. Tsirtsis e P. Srisuresh, *Network address translation - protocol translation (nat-pt)*, RFC 2766 (Historic), Obsoleted by RFC 4966, updated by RFC 3152, Internet Engineering Task Force, fev. de 2000. endereço: http://www.ietf.org/rfc/rfc2766.txt.

[66] S. Turner e L. Chen, *Updated security considerations for the md5 message-digest and the hmac-md5 algorithms*, RFC 6151 (Informational), Internet Engineering Task Force, mar. de 2011. endereço: http://www.ietf.org/rfc/rfc6151.txt.

[67] D. Uckelmann, M. Harrison e F. Michahelles, «An architectural approach towards the future internet of things», em *Architecting the internet of things*, Springer, 2011, pp. 1–24.

[68] C. Wang, S. S. Chow, Q. Wang, K. Ren e W. Lou, «Privacy-preserving public auditing for secure cloud storage», *IEEE Transactions on Computers*, vol. 62, nº 2, pp. 362–375, 2013. DOI: 10.1109/tc.2011.245.

[69] X. Wang, J. Zhang, E. M. Schooler e M. Ion, «Performance evaluation of attribute-based encryption: Toward data privacy in the iot», *2014 IEEE International Conference on Communications (ICC)*, pp. 725–730, 2014. DOI: 10.1109/icc.2014.6883405.

[70] R. H. Weber, «Internet of things – new security and privacy challenges», *Computer Law & Security Review*, vol. 26, nº 1, pp. 23–30, 2010. DOI: 10.1016/j.clsr.2009.11.008.